УДК 519.15

В книге впервые в мировой литературе достаточно полно представлены разнообразные алгоритмы, связанные с нахождением структурных и числовых характеристик объектов из теории графов. В частности, подробно рассматриваются различные алгоритмы поиска решения в задаче коммивояжера. Кроме того, книга содержит большой фактический материал по исследованию потоков в сетях. Многочисленные примеры иллюстрируют работу конкретных алгоритмов. Приводятся оценки сложности соответствующих процедур. Разнообразная тематика и строгое представление алгоритмов сочетаются с доходчивостью изложения.

Книга будет интересна широкому кругу специалистов, сталкивающихся с теорией графов и ее приложениями. Она доступна студентам университетов и втузов соответствующих специальностей.

Редакция литературы по математическим наукам

Copyright © 1975 by
Academic Press Inc. (London)Ltd.
Second printing 1977

К $\frac{20204\text{-}024}{041\ (01)\text{-}78}$ 24-78

© Перевод на русский язык, «Мир», 1978

Предисловие редактора перевода

За последние десять-пятнадцать лет появилось громадное количество работ, в которых демонстрируются возможности языка и методов теории графов не только в математике и традиционных приложениях в химии и электротехнике, но и в социологии, лингвистике, экономике, генетике. Наглядность теоретико-графовых структур и доходчивость языка теории графов позволяют сделать доступными для широкого круга читателей и формулировки довольно сложных прикладных задач, и весьма тонкие методы их решения.

Еще в период становления теории графов в ней возникало немало таких задач, решение которых предполагало построение некоторых алгоритмов (достаточно вспомнить, например, задачу о кёнигсбергских мостах и особенно задачу Гамильтона об обходе вершин додекаэдра). При этом, как правило, не накладывалось никаких ограничений на сложность алгоритмов, а лишь рассматривался вопрос их существования. Внедрение вычислительной техники поставило и перед всей математикой, и перед теорией графов проблему нахождения не произвольных алгоритмов, позволяющих решать те или иные классы задач, а таких алгоритмов, которые допускали бы практическую реализацию с использованием современных вычислительных устройств. Так возникла проблема практической разрешимости задач: найти эффективный или хотя бы достаточно простой в практически важных случаях алгоритм решения задачи. Классификация таких алгоритмов и методов их построения не может быть исчерпывающей. Однако всякое последовательное, систематическое изложение наиболее ярких и ценных алгоритмов, в частности алгоритмов «на графах», и приемов их получения является полезным и даже необходимым для прикладников. В этой связи можно упомянуть книги Ху [1], Корбута и Финкельштейна [2], Цоя и Цхая [3].

В предлагаемой вниманию читателя книге предпринята весьма удачная попытка познакомить тех, кто интересуется прикладной теорией графов, с важными и перспективными алгоритмическими методами теории графов. Книга имеет ярко выраженный прикладной характер. Автор особо выделяет в ней вычислительные и алгоритмические аспекты теории графов. «Чистой» теории гра-

фов в книге мало. Особый интерес представляют главы, посвященные размещению центров и медиан (гл. 5 и 6), задаче коммивояжера (гл. 10) и паросочетаниям (гл. 12).

Автор использовал терминологию Бержа [4]. Мы сочли благоразумным в ряде мест пользоваться терминологией Харари [5].

[1] Ху Т., Целочисленное программирование и потоки в сетях, М., «Мир», 1974.

[2] Корбут А. А. и Финкельштейн Ю. Ю., Дискретное программирование, М., «Наука», 1969.

[3] Цой С. и Цхай С. М., Прикладная теория графов, Алма-Ата, «Наука», 1971.

[4] Берж К., Теория графов и ее применения, М., ИЛ, 1962.

[5] Харари Ф., Теория графов, М., «Мир», 1973.

Г. Гаврилов

1978 г.

Предисловие

Часто бывает полезно и наглядно изображать некоторую ситуацию в виде рисунка, состоящего из точек (вершин), представляющих основные элементы ситуации, и линий (ребер), соединяющих определенные пары этих вершин и представляющих связи между ними. Такие рисунки известны под общим названием *графов*, и эта книга посвящена их изучению. Графы встречаются во многих областях под разными названиями: «структуры» в гражданском строительстве, «сети» в электротехнике, «социограммы» в социологии и экономике, «молекулярные структуры» в химии, «дорожные карты», электрические или газовые «распределительные сети» и так далее.

Благодаря своему широкому применению теория графов в последние годы интенсивно развивается. В большой мере этому способствует также прогресс в области развития больших быстродействующих вычислительных машин. Непосредственное и детальное представление практических систем, таких, как распределительные сети и системы связи, приводит к графам большого размера, успешный анализ которых зависит в равной степени как от существования «хороших» алгоритмов, так и от возможности использования быстродействующих вычислительных машин. Поэтому в настоящей книге основное внимание сосредоточено на разработке и описании алгоритмов анализа графов, хотя при этом часто упоминаются области их приложения, что позволяет максимально приблизить текст книги к решениям практических задач. Надеюсь, что благодаря этому читателю будет легче связать изложенные в книге основные понятия со своей областью деятельности и применить их к разработке новых методов решения своих специфических задач.

Хотя вообще считается, что эффективность алгоритмов имеет самое важное значение, автор не сводит свою книгу к справочнику эффективных алгоритмов. Часто метод рассматривается и по той причине, что он тесно связан с только что введенными понятиями (или получен из них) и ему отдается предпочтение, хотя имеются и другие алгоритмы, столь же (или даже порой несколько более) эффективные.

Определяющим для нашего изложения явилось желание создать у читателя цельное и логически стройное представление об алгоритмах анализа графов.

Название «Теория графов» на самом деле не может подходить ни к какому однотомному изданию, ибо в таком ограниченном объеме нельзя дать сколько-нибудь полного представления об этом предмете. Настоящая книга не является исключением, ее содержание отражает, как это и должно быть интересы автора и его работы в области исследования операций и в вычислительной математике и теории управления.

В главе 2 обсуждаются основные свойства достижимости и связности графов, вопросы построения сильных компонент и баз и их приложение к образованию группировок правления и коалиций в организациях.

В главе 3 рассматриваются две задачи, связанные с выбором экстремальных множеств вершин с заданными свойствами. Обсуждаются способы решения задач об определении максимальных независимых и минимальных доминирующих множеств, последняя обобщается до задачи о покрывающих множествах. Рассматриваются приложения задачи о покрывающих множествах к составлению расписаний вылета самолетов и работы транспорта, к задаче об информационном поиске, обсуждаются вопросы сведения других задач теории графов к задаче о покрывающих множествах.

В главе 4 задача о раскраске вершин рассматривается как особый случай задачи о покрывающих множествах, изучавшейся в предыдущей главе. Даны приложения задачи раскрашивания к составлению расписаний, размещению ресурсов, а также ее обобщения на задачу доставки.

Следующие две главы посвящены задачам размещения в графах. В главе 5 рассматривается задача размещения мультицентров в графе и ее применение к проблемам размещения на сети дорог всевозможных служб, таких, как пожарные команды, полицейские участки, станции скорой помощи. В главе 6 обсуждается задача размещения мультимедиан в графе и ее приложение к проблемам размещения таких служб, как склады или переключательные пункты в системах доставки товаров или в сетях связи. Глава 5 имеет дело с минимаксной, а глава 6 — с минисуммной задачами размещения.

В главе 7 рассматриваются задачи о деревьях, кратчайших остовах и задача Штейнера, а также приложения этих задач к проектированию электрических и газовых распреде ительных сетей.

В главе 8 разбираются задача поиска кратчайшей цепи между парами вершин и ее приложения к задачам выбора маршрута для максимизации пропускной способности или надежности, к специальному случаю сетей ПЕРТ и к анализу критических маршру-

тов. В этой же главе вводится понятие цикла отрицательного веса и демонстрируется применение этого понятия для решения ряда задач теории графов. Кроме того, приводится порядок вычислений вторых, третьих и так далее кратчайших цепей.

Две следующие главы посвящены задачам поиска циклов в графах. В главе 9 рассматриваются общие циклы и разрезы. Здесь же разбираются задача поиска эйлеровых циклов и задача китайского почтальона с ее приложениями к сборке мусора, к доставке молока или почты и к инспектированию систем с распределенными параметрами. Глава 10 состоит из двух частей. В первой рассматривается задача нахождения гамильтонова цикла в неполном графе и ее приложение к машинному планированию. Вторая часть этой главы посвящена задаче поиска кратчайшего гамильтонова цикла, широко известного под названием «задача коммивояжера», и ее приложениям к выбору транспортных маршрутов.

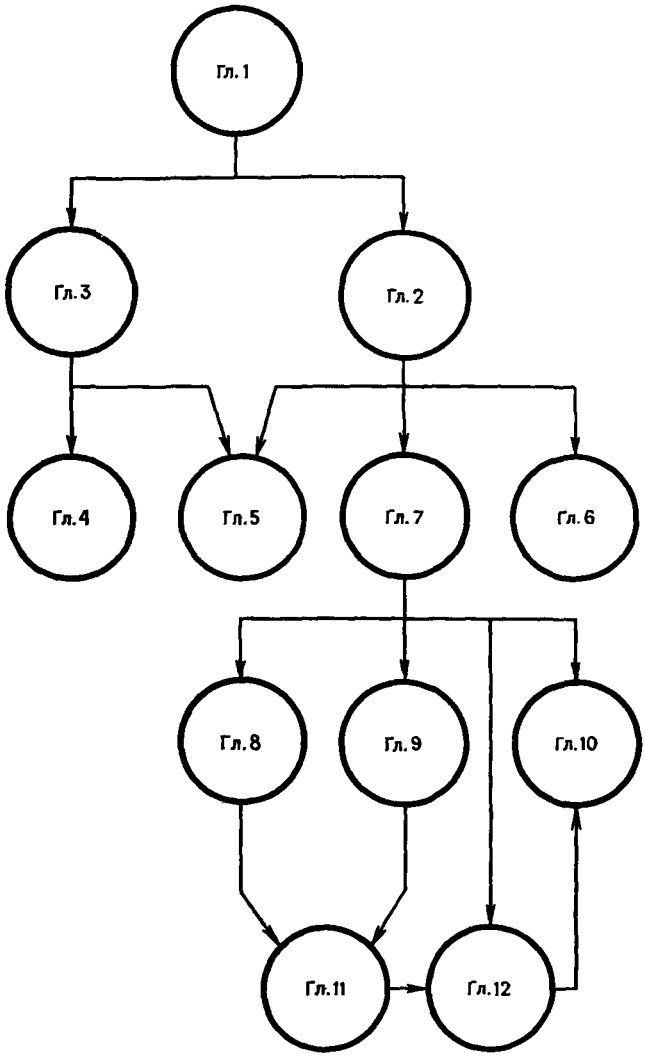
В главе 11 исследуются максимальные потоки и минимальные стоимости — задачи о максимальных потоках в графах, дугам которых приписаны пропускные способности и стоимости. Задачи о потоках в графах, у которых дугам приписаны выигрыши, встречаются при рассмотрении математических моделей арбитража, активных электрических цепей и т. д. Задачи о таких потоках также обсуждаются здесь.

В главе 12 речь идет о вычислении максимальных паросочетаний графов и описывается обобщенный венгерский алгоритм. Подробно алгоритм разобран для двудольных графов, для случаев транспортной задачи и задачи о назначении, играющих большую роль в исследовании операций с приложениями к назначению людей на работу, размещению вспомогательных служб, составлению маршрутов самолетов и т. д. Взаимосвязь между главами этой книги представлена графически (см. стр. 10).

Часть этой книги была написана при финансовой поддержке Научного исследовательского совета по вопросам математического программирования, которому я очень признателен за оказанное содействие. В подготовке издания этой книги мне помогли очень многие. Я особенно благодарен профессору Сэму Эйлону, руководителю Отдела научного управления Империал колледжа, Питеру Виоле, Джеффу Сэлби, Питеру Брукеру и Сэму Кормэну. Я также хочу поблагодарить профессора Дональда Кнута, прочитавшего первый вариант рукописи и указавшего на несколько ошибок. Я в долгу перед г-жой Маргарет Хаджел, взявшей на себя тяжелый труд по перепечатке рукописи. Я должен отметить неоценимую помощь своей жены, которая не только безропотно терпела мою постоянную работу над книгой по вечерам, но также помогала при чтении корректур.

Никос Кристофидес

Октябрь 1973
Лондон



Взаимосвязь между главами книги.

Глава 1

ВВЕДЕНИЕ

1. Графы. Определение

Граф G задается множеством точек или *вершин* $x_1, x_2, \ldots, x_n$ (которое обозначается через X) и множеством линий или *ребер* $a_1, a_2, \ldots, a_m$ (которое обозначается символом A), соединяющих между собой все или часть этих точек. Таким образом, граф G полностью задается (и обозначается) парой (X, A).

Если ребра из множества A ориентированы, что обычно показывается стрелкой, то они называются *дугами*, и граф с такими ребрами называется *ориентированным* графом (рис. 1.1(а)). Если ребра не имеют ориентации, то граф называется *неориентированным* (рис. 1.1(б)). В случае когда $G = (X, A)$ является ориентированным графом и мы хотим пренебречь направленностью дуг из множества A, то неориентированный граф, соответствующий G, будем обозначать как [1]) $\overline{G} = (X, \overline{A})$.

В этой книге, когда дуга обозначается упорядоченной парой, состоящей из *начальной* и *конечной* вершин (т. е. двумя *концевыми* вершинами дуги), ее направление предполагается заданным от первой вершины ко второй [2]). Так, например, на рис. 1.1(а) обозначение (x_1, x_2) относится к дуге a_1, а (x_2, x_1) — к дуге a_2.

Другое, употребляемое чаще описание ориентированного графа G состоит в задании множества вершин X и *соответствия* Γ, которое показывает, как между собой связаны вершины. Соответствие Γ называется *отображением* множества X в X, а граф в этом случае обозначается парой $G = (X, \Gamma)$.

Для графа на рис. 1.1(а) имеем $\Gamma(x_1) = \{x_2, x_5\}$, т. е. вершины x_2 и x_5 являются конечными вершинами дуг, у которых начальной вершиной является x_1.

$$\Gamma(x_2) = \{x_1, x_3\},$$
$$\Gamma(x_3) = \{x_1\},$$
$$\Gamma(x_4) = \varnothing \text{ — пустое множество},$$
$$\Gamma(x_5) = \{x_4\}.$$

---

[1]) Граф $\overline{G}$ мы будем называть *неориентированным дубликатом* (или *неориентированным двойником*) графа G.— *Прим. ред.*

[2]) Если x_1 и x_2 — концевые вершины дуги a, то говорят, что вершины x_1 и x_2 *инцидентны* дуге a (или что дуга a *инцидентна* вершинам x_1 и x_2).— *Прим. ред.*

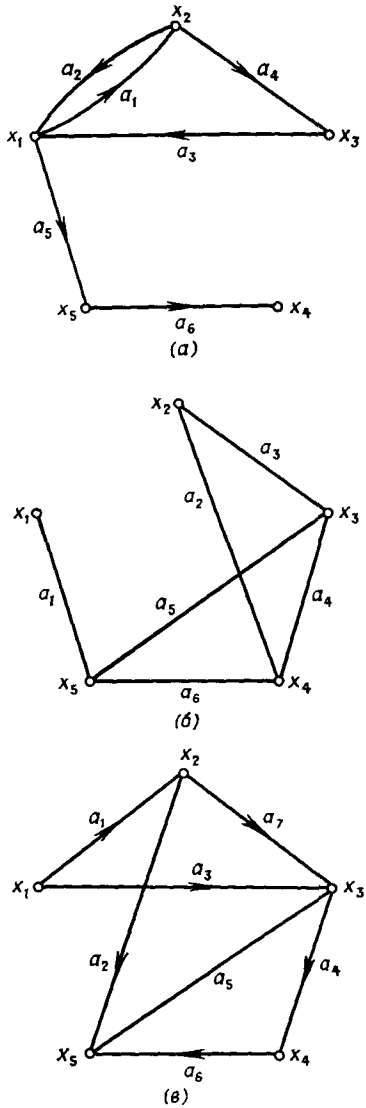


Рис. 1.1. (а) Ориентированный граф. (б) Неориентированный граф. (в) Смешанный граф.

В случае неориентированного графа или графа, содержащего и дуги, и неориентированные ребра (см., например, графы, изображенные на рис. 1.1(б) и 1.1(в)), предполагается, что соответствие Г задает такой эквивалентный ориентированный граф, который

получается из исходного графа заменой каждого неориентированного ребра двумя противоположно направленными дугами, соединяющими те же самые вершины. Так, например, для графа, приведенного на рис. 1.1(б), имеем $\Gamma(x_5) = \{x_1, x_3, x_4\}$, $\Gamma(x_1) = \{x_5\}$ и т. д.

Поскольку $\Gamma(x_i)$ представляет собой множество таких вершин $x_j \in X$, для которых в графе G существует дуга (x_i, x_j), то через $\Gamma^{-1}(x_i)$ естественно обозначить множество вершин x_h, для которых в G существует дуга (x_h, x_i). Отношение $\Gamma^{-1}(x_i)$ принято называть *обратным соответствием*. Для графа, изображенного на рис 1.1(а), имеем

$$\Gamma^{-1}(x_1) = \{x_2, x_3\},$$
$$\Gamma^{-1}(x_2) = \{x_1\}$$

и т. д.

Вполне очевидно, что для неориентированного графа $\Gamma^{-1}(x_i) = \Gamma(x_i)$ для всех $x_i \in X$.

Когда отображение Γ действует не на одну вершину, а на множество вершин $X_q = \{x_1, x_2, \ldots, x_q\}$, то под $\Gamma(X_q)$ понимают объединение

$$\Gamma(x_1) \cup \Gamma(x_2) \cup \ldots \cup \Gamma(x_q),$$

т. е. $\Gamma(X_q)$ является множеством таких вершин $x_j \in X$, что для каждой из них существует дуга (x_i, x_j) в G, где $x_i \in X_q$. Для графа, приведенного на рис. 1.1(а), $\Gamma(\{x_2, x_5\}) = \{x_1, x_3, x_4\}$ и $\Gamma(\{x_1, x_3\}) = \{x_2, x_5, x_1\}$.

Отображение $\Gamma(\Gamma(x_i))$ записывается как $\Gamma^2(x_i)$. Аналогично «тройное» отображение $\Gamma(\Gamma(\Gamma(x_i)))$ записывается как $\Gamma^3(x_i)$ и т. д. Для графа, показанного на рис. 1.1(а), имеем:

$$\Gamma^2(x_1) = \Gamma(\Gamma(x_1)) = \Gamma(\{x_2, x_5\}) = \{x_1, x_3, x_4\},$$
$$\Gamma^3(x_1) = \Gamma(\Gamma^2(x_1)) = \Gamma(\{x_1, x_3, x_4\}) = \{x_1, x_2, x_5\}$$

и т. д.

Аналогично понимаются обозначения $\Gamma^{-2}(x_i)$, $\Gamma^{-3}(x_i)$ и т. д

2. Пути и маршруты

Путем (или *ориентированным маршрутом*) ориентированного графа называется последовательность дуг, в которой конечная вершина всякой дуги, отличной от последней, является начальной вершиной следующей.

Так, на рис. 1.2 последовательности дуг

$$a_6, a_5, a_9, a_8, a_4, \tag{1.1}$$
$$a_1, a_6, a_5, a_9, \tag{1.2}$$
$$a_1, a_6, a_5, a_9, a_{10}, a_6, a_4 \tag{1 3}$$

являются путями.

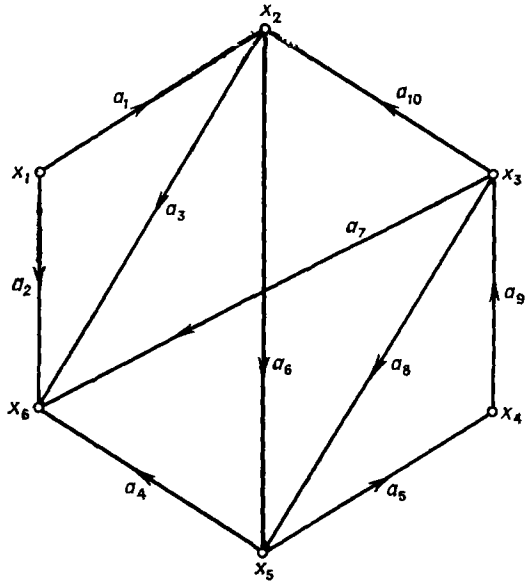


Рис. 1.2.

Дуги $a = (x_i, x_j)$, $x_i \neq x_j$, имеющие общие концевые вершины, называются *смежными*. Две вершины x_i и x_j называются *смежными*, если какая-нибудь из двух дуг (x_i, x_j) и (x_j, x_i) или обе одновременно присутствуют в графе. Так, например, на рис. 1.2 дуги a_1, a_{10}, a_3 и a_6, как и вершины x_5 и x_3, являются смежными, в то время как дуги a_1 и a_5 или вершины x_1 и x_4 не являются смежными.

Ориентированной цепью [1]) (или, короче, *орцепью*) называется такой путь, в котором каждая дуга используется не больше одного раза. Так, например, приведенные выше пути (1.1) и (1.2) являются орцепями, а путь (1.3) не является таким, поскольку дуга a_6 в нем используется дважды.

Простой орцепью [2]) называется такой путь, в котором каждая вершина используется не более одного раза. Например, путь (1.2) является простой орцепью, а пути (1.1) и (1.3) — нет. Очевидно, что простая орцепь является также орцепью, но обратное утверждение неверно. Например, путь (1.1) является орцепью, но не простой орцепью, путь (1.2) является орцепью и простой орцепью, а путь (1.3) не является ни орцепью, ни простой орцепью. *Маршрут* есть неориентированный «двойник» пути, и это понятие рассматривается в тех случаях, когда можно пренебречь направ-

[1]) В оригинале используется термин «простой путь».— *Прим. ред.*
[2]) В оригинале применяется термин «элементарный путь».— *Прим. ред.*

ленностью дуг в графе. Таким образом, *маршрут* есть последовательность ребер $\bar{a}_1, \bar{a}_2, \ldots, \bar{a}_q$, в которой каждое ребро $\bar{a}_i$, за исключением, возможно, первого и последнего ребер, связано с ребрами $\bar{a}_{i-1}$ и $\bar{a}_{i+1}$ своими двумя концевыми вершинами. Последовательности дуг

$$\bar{a}_2, \bar{a}_4, \bar{a}_8, \bar{a}_{10}, \tag{1.4}$$

$$\bar{a}_2, \bar{a}_7, \bar{a}_8, \bar{a}_4, \bar{a}_3 \tag{1.5}$$

и

$$\bar{a}_{10}, \bar{a}_7, \bar{a}_4, \bar{a}_8, \bar{a}_7, \bar{a}_2 \tag{1.6}$$

в графе, изображенном на рис. 1.2, являются маршрутами; черта над символом дуги означает, что ее ориентацией пренебрегают, т. е. дуга рассматривается как неориентированное ребро.

Точно так же, как мы определили орцепи и простые орцепи, можно определить цепи [1]) и простые цепи [2]). Так, например, маршрут (1.4) есть цепь и простая цепь, маршрут (1.5) — цепь, но не простая цепь, а маршрут (1.6) не является ни цепью, ни простой цепью.

Путь или маршрут можно изображать также последовательностью вершин, что мы и будем использовать. Путь (1.1) можно представить также последовательностью вершин x_2, x_5, x_4, x_3, x_5, x_6 и такое представление часто оказывается более полезным в тех случаях, когда осуществляется поиск простых орцепей или простых цепей.

2.1. Веса и длина пути

Иногда дугам графа G сопоставляются (приписываются) числа — дуге (x_i, x_j) ставится в соответствие некоторое число c_{ij}, называемое *весом*, или *длиной*, или *стоимостью* (*ценой*) дуги. Тогда граф G называется *графом со взвешенными дугами*. Иногда веса (числа v_i) приписываются вершинам x_i графа, и тогда получается граф *со взвешенными вершинами*. Если в графе веса приписаны и дугам, и вершинам, то он называется просто *взвешенным* [3]).

При рассмотрении пути μ, представленного последовательностью дуг $(a_1, a_2, \ldots, a_q)$, за его *вес* (или *длину*, или *стоимость*) принимается число $l(\mu)$, равное сумме весов всех дуг [4]), входящих

---

[1]) В оригинале используется термин «простой маршрут».— *Прим. ред.*

[2]) В оригинале применяется термин «элементарный маршрут».— *Прим. ред.*

[3]) Чаще *взвешенным* называют граф со *взвешенными вершинами*.— *Прим. ред.*

[4]) Каждая дуга рассматривается столько раз, сколько она встречается в данном пути.— *Прим. ред.*

в μ, т. е.

$$l(\mu) = \sum_{(x_i, x_j) \in \mu} c_{ij}.$$

Таким образом, когда слова «длина», «стоимость», «цена» и «вес» применяются к дугам, то они эквивалентны по содержанию, и в каждом конкретном случае выбирается такое слово, которое

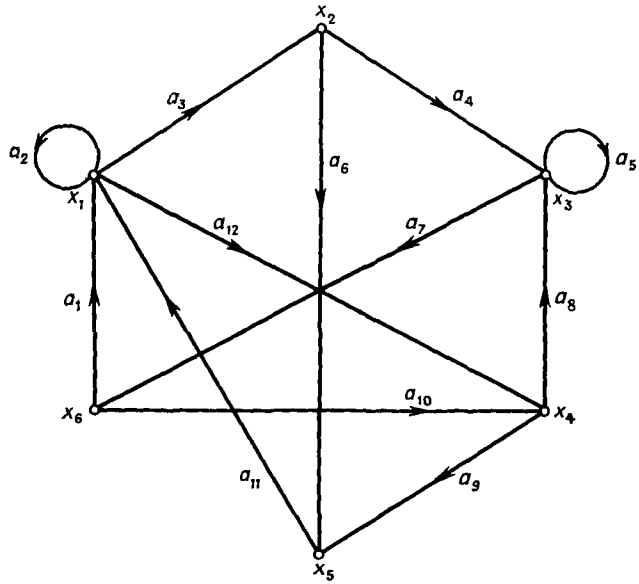


Рис. 1.3.

ближе подходит по смыслу и совпадает с принятым в литературе [1]).

Длиной (или *мощностью*) пути μ называется число дуг, входящих в него [2]).

3. Петли, ориентированные циклы и циклы

Петлей называется дуга, начальная и конечная вершины которой совпадают. На рис. 1.3, например, дуги a_2 и a_5 являются петлями.

Путь $a_1, a_2, \ldots, a_q$ называется *замкнутым*, если в нем начальная вершина дуги a_1 совпадает с конечной вершиной дуги a_q.

[1]) В данном переводе мы применяем термин «длина» только в том случае, когда все дуги, входящие в путь μ, имеют веса, равные 1, т. е. когда вес пути совпадает с его *длиной* (*мощностью*).— *Прим. ред.*

[2]) Каждая дуга считается столько раз, сколько она входит в данный путь.— *Прим. ред.*

Так, например, в графе, приведенном на рис. 1.3, последовательности дуг

$$a_3, a_6, a_{11}, \tag{1.7}$$

$$a_{11}, a_3, a_4, a_7, a_1, a_{12}, a_9, \tag{1.8}$$

$$a_3, a_4, a_7, a_{10}, a_9, a_{11} \tag{1.9}$$

являются замкнутыми путями.

Пути (1.7) и (1.9) являются замкнутыми простыми орцепями (*контурами*, или *простыми орциклами*), поскольку в них одна и та же вершина используется только один раз (за исключением начальной и конечной вершин, которые совпадают), а путь (1.8) не является *контуром*, так как вершина x_1 используется в нем дважды [1]). Контур, проходящий через все вершины графа, имеет особое значение и называется *гамильтоновым* контуром [2]). Конечно, не все графы обладают гамильтоновыми контурами. Так, например, контур (1.9) является гамильтоновым контуром графа, приведенного на рис. 1.3, а граф на рис. 1.2 не имеет гамильтоновых контуров, поскольку не существует такой дуги, для которой x_1 была бы конечной вершиной.

Замкнутый маршрут является неориентированным двойником замкнутого пути. Таким образом, замкнутый маршрут является маршрутом $x_1, x_2, \ldots, x_q$, в котором совпадают начальная и конечная вершины, т. е. в котором $x_1 = x_q$.

На рис. 1.3 маршруты

$$\bar{a}_1, \bar{a}_{12}, \bar{a}_{10} \tag{1.10}$$

и

$$\bar{a}_{10}, \bar{a}_1, \bar{a}_3, \bar{a}_4, \bar{a}_7, \bar{a}_1, \bar{a}_{12} \tag{1.11}$$

являются замкнутыми маршрутами.

---

[1]) Замкнутый путь, в котором нет одинаковых дуг, но, вообще говоря, могут «повторяться» внутренние вершины пути (а не только первая и последняя вершины), т. е. замкнутая орцепь, в оригинале данной книги называется *простым замкнутым* путем. Если же в замкнутом пути нет ни одинаковых дуг, ни одинаковых вершин (кроме первой и последней), то такой путь автор называет *элементарным замкнутым* путем. В нашем переводе мы будем называть замкнутую орцепь *ориентированным циклом* (или, короче, *орициклом*), а элементарный замкнутый путь — *простым орициклом*, или *контуром*.— *Прим. ред.*

[2]) Аналогично определяется *гамильтонов цикл*: это простой цикл, содержащий все вершины графа. Когда нет опасности возникновения недоразумений, мы используем термин «гамильтонов цикл» и в случае ориентированных графов.— *Прим. ред.*

4. Степени вершины

Число дуг, которые имеют вершину x_i своей начальной вершиной, называется *полустепенью исхода* вершины x_i, и, аналогично, число дуг, которые имеют x_i своей конечной вершиной, называется *полустепенью захода* вершины x_i.

Таким образом, на рис. 1.3 полустепень исхода вершины x_6, обозначаемая через $d_0(x_6)$, равна $|\Gamma(x_6)| = 2$, и полустепень захода вершины x_6, обозначаемая через $d_i(x_6)$, равна $|\Gamma^{-1}(x_6)| = 1$.

Совершенно очевидно, что сумма полустепеней захода всех вершин графа, а также сумма полустепеней исхода всех вершин равны общему числу дуг графа G, т. е.

$$\sum_{i=1}^{n} d_0(x_i) = \sum_{i=1}^{n} d_i(x_i) = m, \tag{1.12}$$

где n — число вершин и m — число дуг графа G.

Для неориентированного графа $G = (X, \Gamma)$ *степень* вершины x_i определяется аналогично — с помощью соотношения $d(x_i) \equiv |\Gamma(x_i)|$, и когда не может возникнуть недоразумений, мы будем обозначать степень вершины x_i через d_i.

5. Подграфы

Пусть дан граф $G = (X, A)$. *Остовным подграфом* [1]) G_p графа G называется граф (X, A_p), для которого $A_p \subset A$. Таким образом, остовный подграф имеет то же самое множество вершин, что и граф G, но множество дуг подграфа G_p является подмножеством множества дуг исходного графа.

Граф на рис. 1.4(б) — остовный подграф G_p графа G, изображенного на рис. 1.4(а).

Пусть дан граф $G = (X, \Gamma)$. *Порожденным подграфом* [2]) G_s называется граф (X_s, Γ_s), для которого $X_s \subset X$ и для каждой вершины $x_i \in X_s$, $\Gamma_s(x_i) = \Gamma(x_i) \cap X_s$. Таким образом, порожденный подграф состоит из подмножества вершин X_s множества вершин исходного графа и всех таких дуг графа G, у которых конечные и начальные вершины принадлежат подмножеству X_s. Часто бывает удобно обозначать подграф G_s просто символом $\langle X_s \rangle$; мы будем в дальнейшем использовать такое обозначение, если нет опасности внесения путаницы.

На рис. 1.4(в) показан порожденный подграф графа, приведенного на рис. 1.4(а), содержащий только вершины x_1, x_2, x_3 и x_4 и дуги, которые их связывают.

---

[1]) Автор применяет термин «*частичный граф*».— *Прим. ред.*

[2]) Автор использует термин «подграф».— *Прим. ред.*

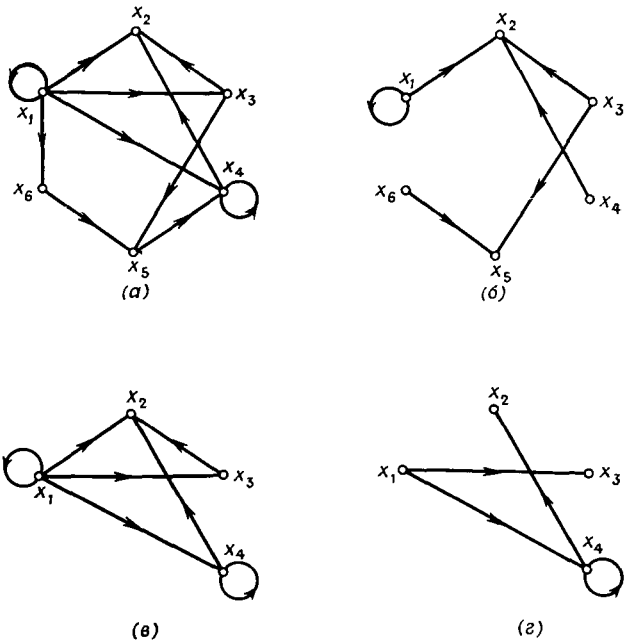


Рис. 1.4. (а) Граф. (б) Остовный подграф.
(в) Порожденный подграф. (г) Подграф.

Соединяя приведенные выше два определения, можно сформулировать определение *подграфа* [1]). Граф, показанный на рис. 1.4(г), является подграфом графа, приведенного на рис. 1.4(а).

Рассмотрим граф, вершины которого представляют сотрудников некоторого учреждения, а дуги — линии связи между сотрудниками. Тогда граф, представляющий только наиболее важные каналы связи данного учреждения, является остовным подграфом; граф, который подробно представляет линии связи только какой-то части этого учреждения (например, отделения), является порожденным подграфом, а граф, который представляет только важные линии связи в пределах отделения, является подграфом.

6. Типы графов

Граф $G = (X, A)$ называют *полным*, если для любой пары вершин x_i и x_j в X существует ребро $(\overline{x_i, x_j})$ в $\bar{G} = (X, \bar{A})$, т. е. для каждой пары вершин графа G должна существовать по край-

[1]) В оригинале используется термин «*частичный подграф*».— *Прим. ред.*

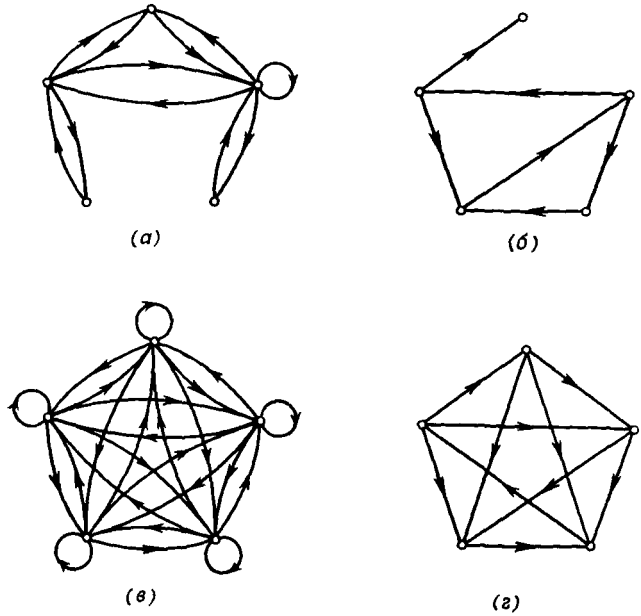


Рис. 1.5. (а) Симметрический граф. (б) Антисимметрический граф. (в) Полный симметрический граф. (г) Полный антисимметрический граф.

ней мере одна дуга, соединяющая их. Полный неориентированный граф, построенный на n вершинах, обозначается через K_n.

Граф (X, A) называется *симметрическим*, если в множестве дуг A для любой дуги (x_i, x_j) существует также противоположно ориентированная дуга (x_j, x_i).

Антисимметрическим графом [1]) называется такой граф, для которого справедливо следующее условие: если $(x_i, x_j) \in A$, то в множестве A нет противоположно ориентированной дуги, т. е. $(x_j, x_i) \notin A$. Очевидно, что в антисимметрическом графе нет петель.

На рис. 1.5(а) показан симметрический граф, а на рис. 1.5 (б) — антисимметрический граф.

Рассмотрим следующий пример: множество вершин графа представляет группу людей, дуга, направленная от вершины x_i к вершине x_j, означает, что x_i является другом или родственником x_j; тогда данный граф должен быть симметрическим. С другой стороны, если дуга, направленная от x_i к x_j, означает, что вершина x_j подчинена вершине x_i, то такой граф должен быть антисимметрическим.

[1]) Применяется также термин «*направленный* граф».— *Прим. ред.*

Комбинируя приведенные выше определения, можно дать определения *полного симметрического* графа (пример такого графа см. на рис. 1.5(в)) и *полного антисимметрического* графа (один из таких графов показан на рис. 1.5(г)). Граф последнего типа часто называют также *турниром*.

Неориентированный граф $G = (X, A)$ называют *двудольным*, если множество его вершин X может быть разбито на такие два подмножества X^a и X^b, что каждое ребро имеет один конец в X^a, а другой в X^b. Ориентированный граф G называется *двудольным*, если его неориентированный двойник $\overline{G}$ — двудольный граф. Легко доказать следующее утверждение.

Теорема 1. *Неориентированный граф G является двудольным тогда и только тогда, когда он не содержит циклов нечетной длины.*

Доказательство. Необходимость. Поскольку X разбивается на две части X^a и X^b, то

$$X^a \cup X^b = X \text{ и } X^a \cap X^b = \varnothing. \tag{1.13}$$

Пусть существует цикл нечетной длины $x_{i_1}, x_{i_2}, \ldots, x_{i_q}, x_{i_1}$, и, без потери общности, допустим, что $x_{i_1} \in X^a$. Поскольку (согласно определению) одна из двух следующих друг за другом вершин этого цикла должна принадлежать X^a, а другая X^b, то имеем $x_{i_2} \in X^b$, $x_{i_3} \in X^a$ и т. д. Следовательно, $x_{i_k} \in X^a$, если k — нечетное, и $x_{i_k} \in X^b$, если k — четное. Мы предположили, что длина цикла нечетная. Поэтому из соотношения $x_{i_q} \in X^a$ следует, что $x_{i_1} \in X^b$. Это противоречит (1.13), поскольку $X^a \cap X^b = \varnothing$ и вершина не может одновременно принадлежать как X^a, так и X^b.

Достаточность. Предположим, что в графе G не существует цикла нечетной длины. Выберем одну из вершин, например x_i, и пометим ее плюсом «+». Выполним следующую итерационную процедуру.

Берем уже помеченную вершину x_i и помечаем все вершины из множества $\Gamma(x_i)$ знаком, противоположным тому, который присвоен вершине x_i.

Будем продолжать эту операцию до тех пор, пока или

(i) все вершины не будут помечены, а знаки, приписанные им, согласованы (иными словами, любые две вершины, соединенные ребром, помечены противоположными знаками), или

(ii) некоторая вершина, например x_{i_k}, которая была уже помечена каким-то знаком («+» или «—»), может быть помечена теперь (со стороны другой вершины) знаком, противоположным приписанному вершине x_{i_k}, или

(iii) для каждой помеченной вершины x_i все вершины из множества $\Gamma(x_i)$ уже помечены, но существуют другие, еще не помеченные вершины.

В случае (i) все вершины, помеченные знаком «+», отнесем к множеству X^a, а помеченные знаком «—» — к множеству X^b. Поскольку все ребра соединяют вершины, помеченные противоположными знаками, то граф является двудольным.

В случае (ii) вершина x_{i_k} должна быть помечена знаком «+» на некотором маршруте (например, μ_1), состоящем из вершин $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$; причем знаки «+» и «—», приписываемые этим вершинам при движении по маршруту μ_1 (от вершины x_{i_1} к вершине x_{i_k}), должны образовывать чередующуюся последовательность (вида +, —, +, ... или —, +, —, ...). Аналогично знаком «—» вершина x_{i_k} помечается вдоль некоторого маршрута μ_2. Пусть x^* — предпоследняя (последней является x_{i_k}) общая вершина маршрутов μ_1 и μ_2. Если вершина x^* помечена знаком «+», то участок от x^* до x_{i_k} маршрута μ_1 должен быть четным, а участок от x^* до x_{i_k} маршрута μ_2 должен быть нечетным. Если же вершина x^* помечена знаком «—», то участок маршрута μ_1 будет нечетным, а маршрута μ_2 — четным. Следовательно, цикл, состоящий из участка маршрута μ_1, от x^* до x_{i_k}, и соответствующего участка маршрута μ_2, от x_{i_k} до x^*, имеет нечетную длину. Это противоречит предположению, что граф G не содержит циклов нечетной длины, и, значит, случай (ii) невозможен.

Случай (iii) означает, что между помеченной и не помеченной вершинами не существует ребра, т. е. что граф G распадается на две или больше частей, и каждая из них может тогда рассматриваться отдельно. Итак, в конце концов приходим к случаю (i) Теорема доказана.

Если нужно подчеркнуть, что граф является двудольным, то для графа применяют обозначение $(X^a \cup X^b, A)$, подразумевая, что выполняются также соотношения (1.13).

Двудольный граф $G = (X^a \cup X^b, A)$ называют *полным*, если для любых двух вершин $x_i \in X^a$ и $x_j \in X^b$ существует ребро (x_i, x_j) в $\bar{G} = (X, \bar{A})$. Если $|X^a|$ — число вершин множества X^a — равно r и $|X^b| = s$, то полный неориентированный двудольный граф $G = (X^a \cup X^b, A)$ обозначается через $K_{r,s}$.

Граф $G = (X, A)$ называется *планарным*, если он может быть нарисован на плоскости (или сфере) таким образом, что произвольные две дуги графа не пересекаются друг с другом [1]). На рис. 1.6(а)

[1]) То есть если точка плоскости (или сферы) принадлежит нескольким дугам графа (двум или большему числу), то она не является внутренней точкой никакой дуги графа.— *Прим. ред.*

показан полный граф K_5, а на рис. 1.6(б) — полный двудольный граф $K_{3,3}$, которые, как известно, являются *непланарными* [1, 3].

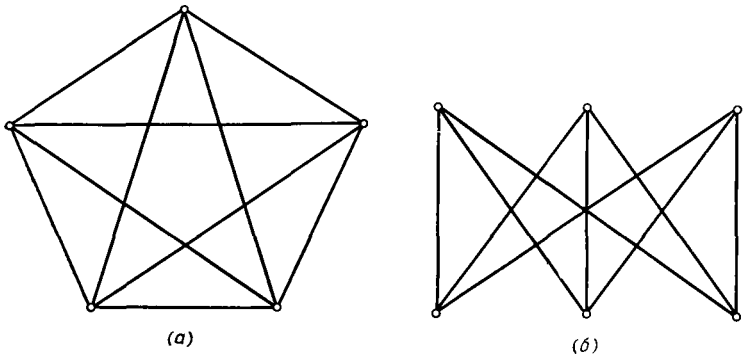


Рис. 1.6. Непланарные графы Куратовского. (а) K_5. (б) $K_{3,3}$.

Эти два графа играют важную роль в теории планарных графов и известны как графы Куратовского.

7. Сильно связные графы и компоненты графа

Ориентированный граф называется *сильно связным* или *сильным*, если для любых двух различных вершин x_i и x_j существует по крайней мере один путь, соединяющий x_i с x_j. Это определение означает также, что любые две вершины такого графа *взаимно достижимы*.

Ориентированный граф называется *односторонне связным* или *односторонним*, если для любых двух различных вершин x_i и x_j существует по крайней мере один путь из x_i в x_j или из x_j в x_i (или оба одновременно).

Ориентированный граф называют *слабо связным* или *слабым*, если для любых двух различных вершин графа существует по крайней мере один маршрут, соединяющий их.

Если для некоторой пары вершин орграфа не существует маршрута, соединяющего их, то такой орграф называется *несвязным* [2].

Граф, приведенный на рис. 1.7(а), как легко проверить, сильно связный. Граф, показанный на рис. 1.7(б), не является сильным (так как в нем нет пути из x_1 в x_3), но односторонне связный. Граф, изображенный на рис. 1.7(в), не является ни сильным, ни односторонним, поскольку в нем не существует путей от x_2 к x_5 и от x_5 к x_2. Он — слабо связный. Наконец, граф, приведенный на рис. 1.7(г), является несвязным.

Пусть дано некоторое свойство P, которым могут обладать графы. *Максимальным подграфом* графа G относительно свойства

P называется порожденный подграф $\langle \hat{X}_s \rangle$ графа G, обладающий этим свойством и такой, что не существует другого порожденного подграфа $\langle X_s \rangle$, у которого $X_s \supset \hat{X}_s$ и который также обладает свойством P. Так, например, если в качестве свойства P взята

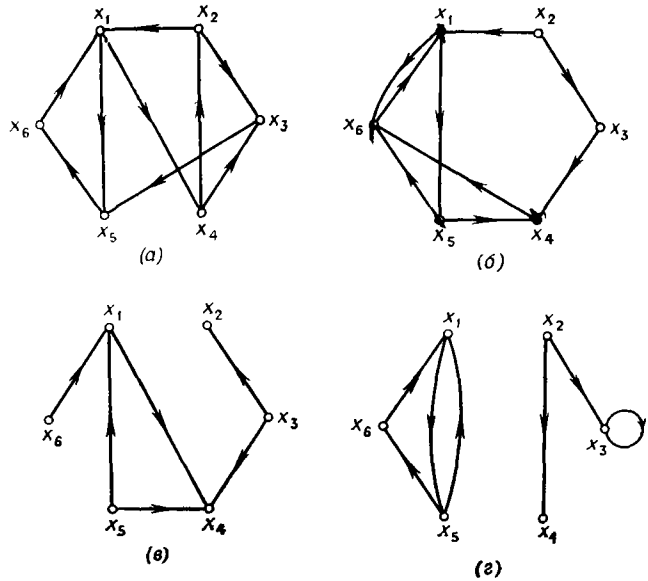


Рис. 1.7. (а) Сильно связный граф. (б) Односторонне связный граф. (в) Слабо связный граф. (г) Несвязный граф.

сильная связность, то максимальным сильным подграфом графа G является сильный подграф, который не содержится в любом другом сильном подграфе. Такой подграф называется *сильной компонентой* графа G. Аналогично, *односторонняя компонента* представляет собой односторонний максимальный подграф, а *слабая компонента* — максимальный слабый подграф.

Например, в графе G, приведенном на рис. 1.7(б), подграф $\langle \{x_1, x_4, x_5, x_6\} \rangle$ является сильной компонентой графа G. С другой стороны, подграфы $\langle \{x_1, x_6\} \rangle$ и $\langle \{x_1, x_5, x_6\} \rangle$ не являются сильными компонентами (хотя и являются сильными подграфами), поскольку они содержатся в графе $\langle \{x_1, x_4, x_5, x_6\} \rangle$ и, следовательно, не максимальные. В графе, показанном на рис. 1.7(в), подграф $\langle \{x_1, x_4, x_5\} \rangle$ является односторонней компонентой. В графе, приведенном на рис. 1.7(г), оба подграфа $\langle \{x_1, x_5, x_6\} \rangle$ и $\langle \{x_2, x_3, x_4\} \rangle$ являются слабыми компонентами, и у этого графа только две такие компоненты.

Из определений сразу же следует, что односторонние компоненты графа могут иметь общие вершины. Сильная компонента должна содержаться по крайней мере в одной односторонней компоненте, а односторонняя компонента содержится в некоторой слабой компоненте данного графа G.

8. Матричные представления

Для алгебраического задания графов удобно использовать следующие матрицы.

8.1. Матрица смежности

Пусть дан граф G, его *матрица смежности* обозначается через $A = [a_{ij}]$ и определяется следующим образом:

$a_{ij} = 1$, если в G существует дуга (x_i, x_j),

$a_{ij} = 0$, если в G нет дуги (x_i, x_j).

Таким образом, матрица смежности графа, изображенного на рис. 1.8, имеет вид

$$\mathbf{A} = \begin{array}{c|cccccc} & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ \hline x_1 & 0 & 1 & 1 & 0 & 0 & 0 \\ x_2 & 0 & 1 & 0 & 0 & 1 & 0 \\ x_3 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_4 & 0 & 0 & 1 & 0 & 0 & 0 \\ x_5 & 1 & 0 & 0 & 1 & 0 & 0 \\ x_6 & 1 & 0 & 0 & 0 & 1 & 1 \end{array}$$

Матрица смежности полностью определяет структуру графа. Например, сумма всех элементов строки x_i матрицы дает полустепень исхода вершины x_i, а сумма элементов столбца x_i — полустепень захода вершины x_i. Множество столбцов, имеющих 1 в строке x_i, есть множество $\Gamma(x_i)$, а множество строк, которые имеют 1 в столбце x_i, совпадает с множеством $\Gamma^{-1}(x_i)$.

Возведем матрицу смежности в квадрат. Пусть элемент $a_{ik}^{(2)}$ матрицы $\mathbf{A}^2$ определяется по формуле

$$a_{ik}^{(2)} = \sum_{j=1}^{n} a_{ij} a_{jk}. \tag{1.14}$$

Слагаемое в уравнении (1.14) равно 1 тогда и только тогда, когда оба числа a_{ij} и a_{jk} равны 1, в противном случае оно равно 0.

Поскольку из равенств $a_{ij} = a_{jk} = 1$ следует существование пути длины 2 из вершины x_i к вершине x_k, проходящего через вершину x_j, то $a_{ik}^{(2)}$ равно числу путей длины 2, идущих из x_i в x_k.

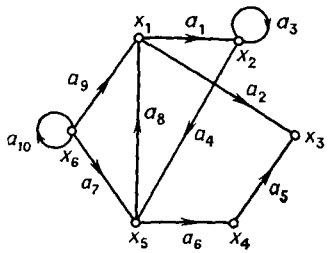


Рис. 1.8.

Аналогично если $a_{ik}^{(p)}$ является элементом матрицы $\mathbf{A}^p$, то $a_{ik}^{(p)}$ равно числу путей (не обязательно орцепей или простых орцепей) длины p, идущих от x_i к x_k.

8.2. Матрица инциденций

Пусть дан граф G с n вершинами и m дугами. *Матрица инциденций* графа G обозначается через $\mathbf{B} = [b_{ij}]$ и является матрицей размерности $n \times m$, определяемой следующим образом:

$b_{ij} = 1$, если x_i является начальной вершиной дуги a_j,

$b_{ij} = -1$, если x_i является конечной вершиной дуги a_j,

$b_{ij} = 0$, если x_i не является концевой вершиной дуги a_j или если a_j является петлей.

Для графа, приведенного на рис. 1.8, матрица инциденций имеет вид

$$\mathbf{B} = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{|cccccccccc|} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 & a_9 & a_{10} \\ \hline 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{array}$$

Поскольку каждая дуга инцидентна двум различным вершинам, за исключением того случая, когда дуга образует петлю, то каж-

дый столбец либо содержит один элемент, равный 1, и один — равный —1, либо все элементы столбца равны 0.

Если G является неориентированным графом, то его матрица инциденций определяется так же, как и выше, за исключением того, что все элементы, равные —1, заменяются на +1.

9. Задачи

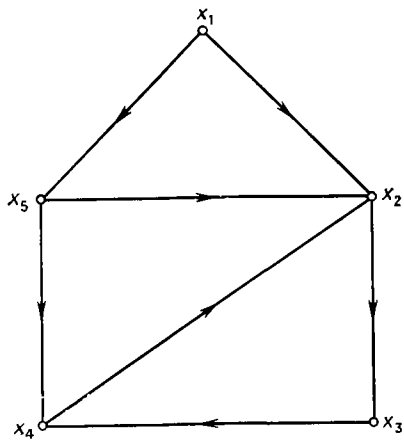


Рис. 1.9.

1. Для графа, приведенного на рис. 1.9, найти
   (а) $\Gamma(x_2)$, (д) $d_0(x_2)$,
   (б) $\Gamma^{-1}(x_2)$, (е) $d_i(x_2)$,
   (в) $\Gamma^2(x_2)$, (ж) матрицу смежности **A**,
   (г) $\Gamma^{-2}(x_2)$, (з) матрицу инциденций **B**.

2. Для графа $G = (X, A)$, изображенного на рис. 1.9, описать
   (а) порожденный подграф $\langle\{x_1, x_2, x_4, x_5\}\rangle$,
   (б) остовный подграф (X, A'), где $(x_i, x_j) \in A'$ тогда и только тогда, когда $i + j$ нечетно,
   (в) остовный подграф подграфа из (а), определенный так же, как в пункте (б).

3. Для неориентированного графа доказать, что в нем число вершин с нечетной степенью четно. (Нуль — четное число.)

4. Показать, что любой полный симметрический граф содержит гамильтонов цикл.

5. Показать, что ранг матрицы инциденций **B** связного графа с n вершинами равен $n-1$; затем доказать, что ранг матрицы инциденций **B** графа с P (слабыми) компонентами равен $n-P$.

6. Доказать, что неориентированный связный граф остается связным после удаления некоторого ребра тогда и только тогда, когда это ребро принадлежит какому-нибудь циклу данного графа.

7. Доказать, что неориентированный граф с n вершинами
   (а) содержит по крайней мере $n-1$ ребер,
   (б) если содержит более, чем $n-1$ ребер, то имеет по крайней мере один цикл.

10. Список литературы

1. Берж К. (1962), Теория графов и ее применения, ИЛ, М.
2. Harary F., Norman R. Z., Cartwright D. (1965), Structural models: An introduction to the theory of directed graphs, Wiley, New York.
3. Kuratowski K. (1930), Sur le problème des courbes gauches en topologie, *Fund. Math.*, 15—16, p. 271.
4. Roy B. (1969, 1970), Algèbre moderne et théorie des graphes, Vol. 1 (1969), Vol. 2 (1970), Dunod, Paris.
5*. Харари Ф., Теория графов, М., «Мир», 1973.

Глава 2

ДОСТИЖИМОСТЬ И СВЯЗНОСТЬ

1. Введение

В предыдущей главе отмечалось, что систему связи любой организации можно интерпретировать как граф, в котором люди представлены вершинами, а каналы связи дугами. Естественно при рассмотрении такой системы поставить вопрос, может ли информация от одного лица x_i быть передана другому лицу x_j, т. е. существует ли путь, идущий от вершины x_i к вершине x_j. Если такой путь существует, то говорят, что вершина x_j *достижима* из x_i. Можно интересоваться достижимостью вершины x_j из вершины x_i только на таких путях, длины которых не превосходят заданной величины. Целью этой главы является обсуждение некоторых фундаментальных понятий, касающихся достижимости и свойств связности графов, а также введение ряда весьма важных алгоритмов.

На языке графов, представляющих организации, в настоящей главе рассматриваются следующие вопросы:

(i) Каково наименьшее число сотрудников, для которых каждый другой сотрудник в этой организации может быть достижим?

(ii) Каково наибольшее число сотрудников, которые взаимно достижимы?

(iii) Как между собой связаны вопросы (i) и (ii)?

2. Матрицы достижимостей и контрадостижимостей

Матрица достижимостей $\mathbf{R} = [r_{ij}]$ определяется следующим образом:

$$r_{ij} = \begin{cases} 1, \text{ если вершина } x_j \text{ достижима из } x_i, \\ 0 \text{ в противном случае.} \end{cases}$$

Множество вершин $R(x_i)$ графа G, достижимых из заданной вершины x_i, состоит из таких элементов x_j, для которых (i, j)-й элемент в матрице достижимостей равен 1. Очевидно, что все диагональные элементы в матрице $\mathbf{R}$ равны 1, поскольку каждая вершина достижима из себя самой с помощью пути длины 0.

Поскольку $\Gamma(x_i)$ является множеством таких вершин x_j, которые достижимы из x_i с использованием путей длины 1 (т. е. $\Gamma(x_i)$ — такое множество вершин, для которых в графе существуют дуги (x_i, x_j)) и поскольку $\Gamma(x_j)$ является множеством вершин, достижимых из x_j с помощью путей длины 1, то множество $\Gamma(\Gamma(x_i)) = \Gamma^2(x_i)$ состоит из вершин, достижимых из x_i с использованием путей длины 2. Аналогично $\Gamma^p(x_i)$ является множеством вершин, которые достижимы из x_i с помощью путей длины p.

Так как любая вершина графа G, которая достижима из x_i, должна быть достижима с использованием пути (или путей) длины 0, или 1, или 2, . . ., или p (с некоторым конечным, но, возможно, достаточно большим значением p), то множество вершин, достижимых из x_i, можно представить в виде

$$R(x_i) = \{x_i\} \cup \Gamma(x_i) \cup \Gamma^2(x_i) \cup \ldots \cup \Gamma^p(x_i). \qquad (2.1)$$

Таким образом, множество $R(x_i)$ может быть получено последовательным выполнением (слева направо) операций объединения в соотношении (2.1), до тех пор, пока «текущее» множество не перестанет увеличиваться по размеру при очередной операции объединения. С этого момента последующие операции не будут давать новых членов множеству и, таким образом, будет образовано достижимое множество $R(x_i)$. Число объединений, которое нужно выполнить, зависит от графа, но, очевидно, что число p меньше числа вершин в графе.

Матрицу достижимостей можно построить так. Находим достижимые множества $R(x_i)$ для всех вершин $x_i \in X$ способом, приведенным выше. Положим $r_{ij} = 1$, если $x_j \in R(x_i)$, и $r_{ij} = 0$ в противном случае. Полученная таким образом матрица $\mathbf{R}$ является матрицей достижимостей.

Матрица контрадостижимостей [1]) $\mathbf{Q} = [q_{ij}]$ определяется следующим образом:

$$q_{ij} = \begin{cases} 1, & \text{если из вершины } x_j \text{ можно достигнуть вершину } x_i, \\ 0 & \text{в противном случае.} \end{cases}$$

Контрадостижимым множеством $Q(x_i)$ графа G является множество таких вершин, что из любой вершины этого множества можно достигнуть вершину x_i. Аналогично построению достижимого множества $R(x_i)$ на основе соотношения (2.1) можно «сформировать» множество $Q(x_i)$, используя следующее выражение:

$$Q(x_i) = \{x_i\} \cup \Gamma^{-1}(x_i) \cup \Gamma^{-2}(x_i) \cup \ldots \cup \Gamma^{-p}(x_i), \qquad (2.2)$$

где $\Gamma^{-2}(x_i) = \Gamma^{-1}(\Gamma^{-1}(x_i))$ и т. д.

---

[1]) В оригинале применяется термин «*reaching matrix*». Мы будем использовать также термин «*матрица обратных достижимостей*». — *Прим. ред.*

Операции выполняются слева направо до тех пор, пока очередная операция объединения не перестанет изменять «текущее» множество $Q(x_i)$.

Из определений очевидно, что столбец x_i матрицы $\mathbf{Q}$ (в котором $q_{ij} = 1$, если $x_j \in Q(x_i)$, и $q_{ij} = 0$ в противном случае) совпадает со строкой x_i матрицы $\mathbf{R}$, т. е. $\mathbf{Q} = \mathbf{R}^t$, где $\mathbf{R}^t$ — матрица, транспонированная к матрице достижимостей $\mathbf{R}$.

2.1. Пример

Найти матрицы достижимостей и обратных достижимостей для графа G, приведенного на рис. 2.1. Матрица смежности графа

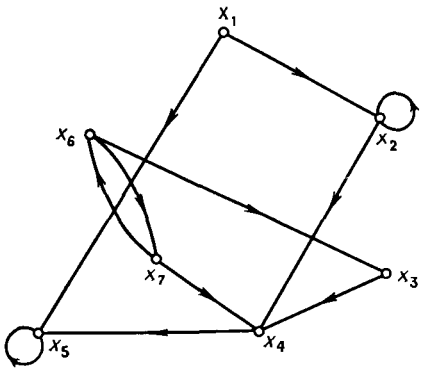


Рис. 2.1.

G имеет вид

$$\mathbf{A} = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Множества достижимостей находятся с помощью соотношения (2.1):

$$R(x_1) = \{x_1\} \cup \{x_2, x_5\} \cup \{x_2, x_4, x_5\} \cup \{x_2, x_4, x_5\} = \\ = \{x_1, x_2, x_4, x_5\},$$

$$R(x_2) = \{x_2\} \cup \{x_2, x_4\} \cup \{x_2, x_4, x_5\} \cup \{x_2, x_4, x_5\} = \{x_2, x_4, x_5\},$$

$$R(x_3) = \{x_3\} \cup \{x_4\} \cup \{x_5\} \cup \{x_5\} = \{x_3, x_4, x_5\},$$

$$R(x_4) = \{x_4\} \cup \{x_5\} \cup \{x_5\} = \{x_4, x_5\},$$

$$R(x_5) = \{x_5\} \cup \{x_5\} = \{x_5\},$$

$$R(x_6) = \{x_6\} \cup \{x_3, x_7\} \cup \{x_4, x_6\} \cup \{x_3, x_5, x_7\} \cup \{x_4, x_5, x_6\} = \{x_3, x_4, x_5, x_6, x_7\},$$

$$R(x_7) = \{x_7\} \cup \{x_4, x_6\} \cup \{x_3, x_5, x_7\} \cup \{x_4, x_5, x_6\} = \{x_3, x_4, x_5, x_6, x_7\}$$

Следовательно, матрица достижимостей имеет вид

$$\mathbf{R} = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

матрица обратных достижимостей такова:

$$\mathbf{Q} = \mathbf{R}' = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

Следует отметить, что поскольку все элементы матриц **R** и **Q** равны 1 или 0, то каждую строку можно хранить в двоичной форме, используя одно (или больше) машинных слов. Таким

образом, нахождение матриц **R** и **Q** с вычислительной точки зрения является довольно простой задачей, поскольку объединение множеств в соответствии с выражениями (2.1) и (2.2) и сравнение «текущих» множеств после каждого объединения, проводимое для выяснения необходимости продолжения процесса построения соответствующих множеств,— все это можно осуществить на ЭВМ с помощью одной логической операции [1]).

Так как $R(x_i)$ является множеством вершин, достижимых из x_i, а $Q(x_j)$ — множеством вершин, из которых можно достигнуть x_j, то $R(x_i) \cap Q(x_j)$ — множество таких вершин, каждая из которых принадлежит по крайней мере одному пути, идущему от x_i к x_j. Эти вершины называются существенными или *неотъемлемыми* относительно двух концевых вершин x_i и x_j [7]. Все остальные вершины $x_k \notin R(x_i) \cap Q(x_j)$ называются несущественными или *избыточными*, поскольку их удаление не влияет на пути от x_i к x_j.

Матрицы достижимостей и обратных достижимостей, определенные выше, являются полными в том смысле, что на длины путей от x_i к x_j не накладывались никакие ограничения. С другой стороны, можно определить матрицы *ограниченных* достижимостей и контрадостижимостей — надо потребовать, чтобы длины путей не превышали некоторого заданного числа. Эти ограниченные матрицы тоже могут быть построены с помощью соотношений (2.1) и (2.2) — надо действовать точно так, как раньше, при нахождении «неограниченных» матриц, но только теперь p будет верхней границей длины допустимых путей.

Граф называют *транзитивным*, если из существования дуг (x_i, x_j) и (x_j, x_k) следует существование дуги (x_i, x_k). *Транзитивным замыканием* графа $G = (X, A)$ является граф $G_{tc} = (X, A \cup A')$, где A' является минимально возможным множеством дуг, необходимых для того, чтобы граф G_{tc} был транзитивным. Так как путь от x_i к x_j в графе G должен соответствовать дуге (x_i, x_j) в G_{tc}, то совершенно очевидно, что матрица достижимостей **R** графа G почти полностью совпадает с матрицей смежности **A** графа G_{tc} — надо только в матрице **A** поставить на главной диагонали единицы.

3. Нахождение сильных компонент

Сильная компонента (СК) графа G была определена в предыдущей главе как максимальный сильно связный подграф графа G. Поскольку в сильно связном графе произвольная вершина x_j достижима из любой другой вершины x_i, то в ориентированном

[1]) В гл. 7, посвященной изучению *деревьев*, приводятся другие способы построения множеств $R(x_i)$ и $Q(x_i)$. Эти способы базируются на процедуре расстановки пометок в вершинах графа (пометки ставят начиная с вершины x_i).

графе существует одна и только одна СК, содержащая данную вершину x_i. В самом деле, если бы вершина x_i принадлежала двум или большему числу сильных компонент, то существовал бы путь из любой вершины одной СК в произвольную вершину другой СК и, следовательно, объединение этих сильных компонент было бы сильно связным графом, что противоречит определению СК.

Если вершина x_i одновременно является начальной и конечной вершиной пути, то множество вершин, существенных относительно этих двух идентичных концов (т. е. множество вершин некоторого цикла, содержащего x_i), совпадает с пересечением $R(x_i) \cap Q(x_i)$. Поскольку все эти существенные вершины достижимы из x_i и, кроме того, из каждой такой вершины достижима вершина x_i, то все они взаимно достижимы. Более того, если нет другой вершины, существенной относительно концов x_i и x_i, то множество $R(x_i) \cap Q(x_i)$, которое может быть построено с использованием соотношений (2.1) и (2.2), однозначно определяет СК графа G, содержащую вершину x_i.

Если эти вершины удалить из графа $G = (X, \Gamma)$, то в оставшемся порожденном подграфе $G' = \langle X - R(x_i) \cap Q(x_i) \rangle$ можно таким же способом выделить новую СК, содержащую $x_j \in X - R(x_i) \cap Q(x_i)$. Эту процедуру можно повторять до тех пор, пока все вершины графа G не будут сгруппированы в соответствующие СК. После завершения этой процедуры граф G будет *разбит* на свои сильные компоненты [3].

Граф $G^* = (X^*, \Gamma^*)$ определяется так: каждая его вершина представляет множество вершин некоторой сильной компоненты [1]) графа G, дуга (x_i^*, x_j^*) существует в G^* тогда и только тогда, когда в G существует дуга (x_i, x_j), такая, что x_i принадлежит компоненте, соответствующей вершине x_i^*, а x_j — компоненте, соответствующей вершине x_j^*. Граф G^* называют *конденсацией* графа G.

Совершенно очевидно, что конденсация G^* не содержит циклов, поскольку наличие цикла означает, что любые вершины этого цикла взаимно достижимы, а поэтому совокупность всех вершин цикла принадлежит некоторой СК в G^* и, следовательно, содержится в СК графа G, что противоречит определению конденсации, в силу которого вершины из G^* соответствуют СК в G.

3.1. Пример

Для графа G, приведенного на рис. 2.2, найти сильные компоненты и построить конденсацию G^*.

Найдем СК в G, содержащую вершину x_1.

[1]) Разным компонентам из G соответствуют разные вершины в графе G^*.— *Прим. ред.*

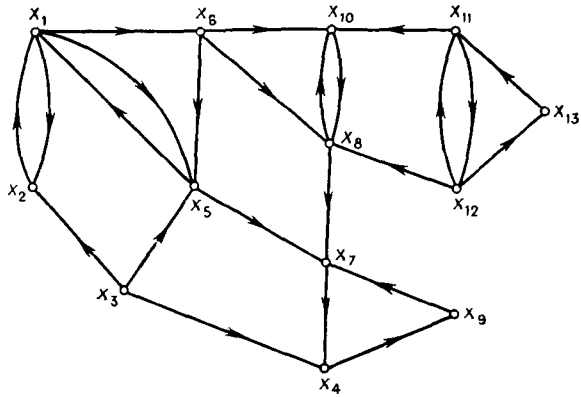


Рис. 2.2. Граф G.

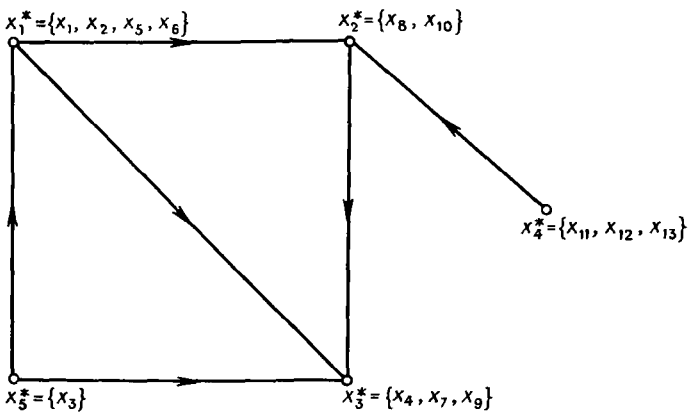


Рис. 2.3. G^* — конденсация графа G.

Из соотношений (2.1) и (2.2) получаем

$$R(x_1) = \{x_1, x_2, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\}$$

и

$$Q(x_1) = \{x_1, x_2, x_3, x_5, x_6\}.$$

Следовательно, СК, содержащая вершину x_1, является порожденным подграфом

$$\langle R(x_1) \cap Q(x_1)\rangle = \langle\{x_1, x_2, x_5, x_6\}\rangle.$$

Аналогично, СК, содержащая вершину x_8, есть порожденный подграф $\langle\{x_8, x_{10}\}\rangle$, СК содержащая x_7 — подграф $\langle\{x_4, x_7, x_9\}\rangle$,

СК, содержащая x_{11},— подграф $\langle\{x_{11}, x_{12}, x_{13}\}\rangle$ и СК, содержащая x_3,— подграф $\langle\{x_3\}\rangle$. Следует отметить, что последняя СК состоит из единственной вершины графа G.

Конденсация G^* приведена на рис. 2.3.

Процедуру, описанную выше и связанную с нахождением СК графа, можно сделать более удобной, если непосредственно использовать матрицы **R** и **Q**, определенные в предшествующем разделе. Пусть запись $\mathbf{R} \otimes \mathbf{Q}$ означает поэлементное умножение этих матриц; тогда сразу видно, что строка x_i матрицы $\mathbf{R} \otimes \mathbf{Q}$ содержит единицы только в тех столбцах x_j, для которых выполняется условие: вершины x_i и x_j взаимно достижимы; в других местах строки x_i стоят нули. Таким образом, две вершины находятся в одной и той же СК тогда и только тогда, когда соответствующие им строки (или столбцы) в матрице $\mathbf{R} \otimes \mathbf{Q}$ идентичны. Вершины, которым соответствуют строки, содержащие 1 в столбце x_j, образуют множество вершин СК, содержащей x_j. Отсюда мгновенно следует, что матрицу $\mathbf{R} \otimes \mathbf{Q}$ можно преобразовать путем транспонирования строк и столбцов в блочно-диагональную ; каждая из диагональных подматриц этой матрицы соответствует СК графа G и содержит только единичные элементы; все остальные элементы блочно-диагональной матрицы равны нулю. Для приведенного ранее примера матрица $\mathbf{R} \otimes \mathbf{Q}$, преобразованная соответствующим образом, имеет вид

$\mathbf{R} \otimes \mathbf{Q} =$

	x_1	x_2	x_5	x_6	x_8	x_{10}	x_4	x_7	x_9	x_{11}	x_{12}	x_{13}	x_3
x_1	1	1	1	1									
x_2	1	1	1	1									
x_5	1	1	1	1	0		0			0			0
x_6	1	1	1	1									
x_8					1	1							
x_{10}					1	1	0			0			0
x_4							1	1	1				
x_7			0		0		1	1	1	0			0
x_9							1	1	1				
x_{11}										1	1	1	
x_{12}			0		0		0			1	1	1	0
x_{13}										1	1	1	
x_3			0		0		0			0			1

4. Базы

База [1]) B графа есть множество вершин, из которого достижима любая вершина графа и которое является минимальным в том смысле, что не существует собственного подмножества в B, обладающего таким свойством достижимости. Если мы обозначим через $R(B)$ множество вершин, достижимых из вершин множества B, т. е.

$$R(B) = \bigcup_{x_i \in B} R(x_i), \tag{2.3}$$

то B является базой тогда и только тогда, когда

$$R(B) = X \qquad \text{и} \qquad \forall S \subset B \, (R(S) \neq X). \tag{2.4}$$

Второе условие ($R(S) \neq X \;\; \forall S \subset B$) в соотношении (2.4) эквивалентно утверждению: $x_j \notin R(x_i)$ для любых двух различных x_i, $x_j \in B$, т. е. вершина из B не достижима из любой другой вершины B. Эта эквивалентность может быть обоснована следующим образом. Поскольку для любых двух множеств H и $H' \subseteq H$ мы имеем, что $R(H') \subseteq R(H)$, то условие $R(S) \neq X \;\; \forall S \subset B$ эквивалентно соотношению $R(B - \{x_j\}) \neq X$ для всех $x_j \in B$; другими словами, $R(x_j) \not\subseteq R(B - \{x_j\})$. Последнее условие может быть выполнено тогда и только тогда, когда $x_j \notin R(B - \{x_j\})$, т. е. тогда и только тогда, когда $x_j \notin R(x_i)$ для любых x_i, $x_j \in B$.

Итак, базой является такое множество B вершин графа G, которое удовлетворяет следующим двум условиям:

(i) каждая вершина графа G достижима хотя бы из одной вершины множества B и

(ii) в B нет вершины, которая достижима из другой вершины множества B.

Из этих двух условий мгновенно получаются следующие утверждения:

(а) в множестве B нет двух вершин, которые принадлежат одной и той же СК графа G,

(б) в любом графе без циклов существует единственная база; она состоит из всех таких вершин графа, полустепени захода которых равны 0.

Доказательства этих двух утверждений простые и непосредственно следуют из определений. (См. задачи 2.3 и 2.4.)

Таким образом, в силу утверждений (а) и (б) база B^* конденсации G^* графа G состоит из таких вершин графа G^*, полустепени

[1]) Или *вершинная база.— Прим. ред.*

захода которых равны 0. Следовательно, базы графа G можно строить так: из каждой СК графа G, соответствующей вершине базы B^* конденсации G^* надо взять по одной вершине, т. е. если $B^* = \{S_1, S_2, \ldots, S_m\}$, где m — число вершин-множеств S_j в базе B^* графа G^*, то базой B является произвольное множество $\{x_{i_1}, x_{i_2}, \ldots, x_{i_m}\}$, где $x_{i_j} \in S_j$.

4.1. Пример

Для графа G, приведенного на рис. 2.2, конденсация G^* показана на рис. 2.3. Базой графа G^* является множество $\{x_4^*, x_5^*\}$, поскольку x_4^* и x_5^* — единственные вершины в G^* с полустепенями захода, равными 0. Базами графа G являются $\{x_3, x_{11}\}$, $\{x_3, x_{12}\}$ и $\{x_3, x_{13}\}$.

Понятие, двойственное понятию базы, можно дать следующим образом на языке контрадостижимых множеств $Q(x_i)$.

Антибаза $\overline{B}$ есть множество вершин графа $G = (X, \Gamma)$, таких, что

$$\left.\begin{array}{l} Q(\overline{B}) = \bigcup\limits_{x_i \in \overline{B}} Q(x_i) = X \\ \text{и} \\ \forall S \subseteq \overline{B},\ Q(S) \neq X, \end{array}\right\} \qquad (2.5)$$

т. е. $\overline{B}$ есть такое минимально возможное множество вершин, что какова бы ни была вершина графа G, из нее достижима некоторая вершина в $\overline{B}$. Свойства антибаз аналогичны свойствам баз, надо только «прямые» понятия заменить на двойственные. Например, соотношения (2.5) эквивалентны двум условиям, подобным (i) и (ii), приведенным выше, необходимо лишь заменить B на $\overline{B}$ и сделать другие «двойственные» преобразования [1]).

Таким образом, антибаза конденсации G^* есть множество вершин в G^*, полустепени исхода которых равны 0, и антибазы самого графа G строятся из антибазы графа G^* путем выбора по одной вершине в каждой вершине-множестве антибазы $\overline{B}^*$ — подобно тому, как это делалось раньше для баз.

В примере с графом G, изображенным на рис. 2.2, конденсация G^* (рис. 2.3) содержит только одну вершину x_3^* с полустепенью исхода, равной 0. Таким образом, антибаза графа G^* есть $\{x_3^*\}$, а антибазами графа G являются множества $\{x_4\}$, $\{x_7\}$ и $\{x_9\}$.

---

[1]) Условие, двойственное условию (i), формулируется, например, так: из любой вершины графа G достижима хотя бы одна вершина множества $\overline{B}$. — *Прим. ред.*

4.2. Применение к исследованию структуры организаций

Если граф G представляет структуру руководства или влияний некоторой организации, то члены каждой сильной компоненты графа G имеют равную власть или равное влияние друг на друга, как это может быть, например, для случая комитета. Базу графа G можно интерпретировать как «коалицию», включающую наименьшее число лиц, обладающих властью над каждым членом организации [2, 3].

Пусть на множестве вершин, представляющих членов той же самой организации, построен новый граф G', отображающий каналы связи, так что каждая дуга (x_i, x_j) означает, что x_i может связываться с x_j. Граф G', конечно, каким-то образом связан с графом G, но совсем не очевидным образом. Наименьшее число лиц, которые знают или могут получить все сведения об организации, образует одну из антибаз графа G'. Можно утверждать, что эффективная для управления этой организацией коалиция будет множеством лиц H, определяемым следующим соотношением:

$$H = B(G) \cup \overline{B}(G'), \tag{2.6}$$

где $B(G)$ и $\overline{B}(G')$ — одна из баз графа G и одна из антибаз графа G', выбранные так, чтобы $|H|$ — число людей в H — было минимальным.

Приведенное выше описание организации с использованием языка теории графов является, конечно, сильно упрощенным. Один из недостатков, который сразу бросается в глаза, состоит в том, что нежелательно, чтобы лицо, не входящее в B, имело бы власть над лицом из B.

Следовательно, можно определить *сильную базу* [1]) как такое множество вершин $B_p \subseteq X$, что

$$R(B_p) = X, \quad Q(B_p) = B_p \tag{2.7а}$$

и

$$R(S) \neq X \qquad \forall S \subset B_p. \tag{2.7б}$$

Вторая часть условия (2.7а) выражает тот факт, что только лица из B_p могут иметь власть над другими лицами, также принадлежащими B_p, и может быть заменено эквивалентным условием $R(X - B_p) \cap B_p = \varnothing$. Это условие означает, что если вершина из СК графа G входит в B_p, то и каждая вершина из той же самой СК должна входить в B_p. Поскольку база в G^* есть множество таких вершин, полустепени захода которых равны 0, т. е. ни одна из этих вершин не достижима ни из какой другой вершины графа, то сильная база в G есть объединение множеств вершин базы гра-

[1]) В оригинале «*power-basis*».— *Прим. ред.*

фа G^*, т. е.

$$B_p = \bigcup_{S_i \in B^*} S_i. \tag{2.8}$$

Для графа, приведенного в примере 4.1 (рис. 2.2 и 2.3), сильная база G есть $\{x_3, x_{11}, x_{12}, x_{13}\}$. Можно отметить, что, если этот граф представляет организацию, то x_3 можно рассматривать как руководителя, обладающего властью над всеми множествами лиц x_1^*, x_2^* и x_3^*, в то время как $\{x_{11}, x_{12}, x_{13}\}$ можно рассматривать как комитет, имеющий власть над двумя множествами лиц x_2^* и x_3^*.

5. Задачи, связанные с ограниченной достижимостью

В предыдущем разделе при определении базы мы исходили из неограниченных достижимых множеств. Если рассматривать ограниченную достижимость, т. е. использовать пути ограниченной длины (как упоминалось ранее в разд. 2), и определять ограниченную базу, опираясь на ограниченную достижимость, то возникают осложнения двух типов.

(а) Понятия сильной компоненты и конденсации, которые упрощают решение задачи определения баз в случае неограниченной достижимости, теперь использоваться не могут. Распространение этих понятий на случай ограниченной достижимости не дает существенного упрощения задачи.

В случае когда достижимость ограничена путями единичной длины (просто дугами), ограниченные базы называют *минимальными доминирующими множествами*. Мы подробно рассматриваем их в гл. 3. В случае когда достижимость ограничена, скажем, q дугами [1]), граф G' может быть определен как граф, множество вершин которого то же самое, что и у графа G, а дуга (x_i, x_j) существует в G' тогда и только тогда, когда в G найдется путь между вершинами x_i и x_j длины, не больше чем q (см. соотношение 2.1). Ограниченная матрица достижимостей графа G соответствует тогда матрице смежности графа G'; граф G' может быть назван ограниченным транзитивным замыканием графа G (в соответствии с определением транзитивного замыкания, приведенным в разд. 2). Задача определения ограниченных баз графа G эквивалентна задаче нахождения минимальных доминирующих множеств графа G'.

(б) В неограниченном случае различные базы имеют одно и то же число элементов; оно равно числу таких вершин в конденсации графа, которые имеют нулевые полустепени захода. В огра-

[1] То есть рассматриваются только пути, длины которых не превосходят q.— *Прим. ред.*

ниченном случае, однако, разные базы могут иметь различное число элементов, и тогда возникает, в частности, задача нахождения базы с наименьшим числом элементов. Еще один подход: если не наложено ограничение на достижимость, то можно искать такую ограниченную базу, которая содержит, например, ровно p вершин и, кроме того, из вершин этой базы все остальные вершины графа G могут быть достигнуты с минимально возможной ограниченностью достижимости. Эти задачи тесно связаны с задачей нахождения *p-центра* и более детально рассматриваются в гл. 4.

6. Задачи

1. Для графа, приведенного на рис. 2.4, найти матрицы достижимостей и контрадостижимостей.

2. Для графа, приведенного на рис. 2.4, найти сильные компоненты, начертить конденсацию и построить все его базы.

3. Доказать, что в любом графе G каждая его база содержит все вершины, имеющие нулевые полустепени захода.

4. Доказать, что в любом графе без циклов имеется только одна база. Она состоит из всех вершин с нулевыми полустепенями захода.

5. Показать, что любые две базы графа G имеют одно и то же число вершин.

6. Доказать, что вершина x_i принадлежит одновременно базе B и антибазе $\overline{B}$ графа G тогда и только тогда, когда сильная компонента, содержащая x_i, соответствует изолированной вершине в конденсации G^*.

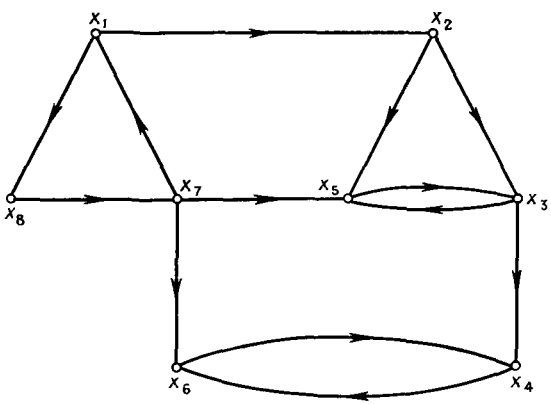


Рис. 2.4.

7. Показать, что если $\hat{x}_i$ есть такая вершина графа (X, A), на которой достигается

$$\max_{x_i \in X} |R(x_i)|,$$

то $\hat{x}_i$ принадлежит базе.

8. Пусть все арифметические операции, рассматриваемые ниже, являются булевскими (т. е. $0+0=0$, $0+1=1+0=1$, $1+1=1$, $0 \cdot 0 = 0 \cdot 1 = 0$ и $1 \cdot 1 = 1$). Показать, что если рассматривать достижимость только на путях длины не больше чем q, то матрица $\mathbf{R}_q$ ограниченных достижимостей дается формулой

$$\mathbf{R}_q = \mathbf{I} + \mathbf{A} + \mathbf{A}^2 + \ldots + \mathbf{A}^q = (\mathbf{I} + \mathbf{A})^q,$$

где $\mathbf{I}$ — единичная $(n \times n)$-матрица.

9. Показать, что элемент $r_{ii}^{(2)}$ матрицы $\mathbf{R}^2$ равен числу вершин в сильной компоненте, содержащей вершину x_i.

7. Список литературы

1. Chen Y. C., Wing O. (1964), Connectivity of directed graphs, Proc. of Allerton Conference on Circuit and System Theory, Univ. of Illinois.
2. Flament C. (1963), Applications of graph theory to group structure, Prentice-Hall, Englewood Cliffs, New Jersey.
3. Harary F., Norman R. Z., Cartwright D. (1965), Structural models: An introduction to the theory of directed graphs, Wiley, New York.
4. Лейфман Л. Я. (1966), Эффективный алгоритм разбиения ориентированного графа на бикомпоненты, *Кибернетика*, 2, стр. 19.
5. Marimont R. B. (1959), A new method of checking the consistency of precedence matrices, *J. of ACM*, 6, p. 164.
6. Moyles D. M., Thompson G. L. (1969), An algorithm for finding a minimum equivalent graph of a digraph, *J. of ACM*, 16, p. 455.
7. Ramamoorthy C. V. (1966), Analysis of graphs by connectivity considerations, *J. of ACM*, 13, p. 211.

Глава 3

НЕЗАВИСИМЫЕ И ДОМИНИРУЮЩИЕ МНОЖЕСТВА. ЗАДАЧА О ПОКРЫВАЮЩИХ МНОЖЕСТВАХ

1. Введение

Пусть дан граф $G = (X, \Gamma)$. Довольно часто возникает задача поиска таких подмножеств множества вершин X графа G, которые обладают определенным, наперед заданным свойством. Например, какова максимально возможная мощность такого подмножества $S \subseteq X$, для которого порожденный подграф $\langle S \rangle$ является полным? Или какова максимальная мощность подмножества S, такого, что граф $\langle S \rangle$ — вполне несвязный? Ответ на первый вопрос дает так называемое *кликовое число* графа G, а на второй — *число независимости* [1]). Еще одна задача. Она состоит в нахождении минимально возможной мощности таких подмножеств S множества X, что любая вершина из $X - S$ достижима из S с помощью путей единичной длины. Решение этой задачи дается так называемым *числом доминирования* графа G.

Эти числа и связанные с ними подмножества вершин описывают важные структурные свойства графа и имеют разнообразные непосредственные приложения при ведении проектного планирования исследовательских работ [3], в кластерном анализе и численных методах таксономии [36, 2], параллельных вычислениях на ЭВМ [17], при размещении предприятий обслуживания, а также источников и потребителей в энергосистемах [66, 56].

В настоящей главе приводятся алгоритмы определения указанных выше чисел и обсуждаются некоторые их приложения. Кроме того, рассматривается *задача о наименьшем покрытии*, которая является обобщением задачи о нахождении числа доминирования графа, и излагается некоторый метод ее решения. Последняя задача очень важна не только потому, что она имеет большое число прямых приложений, но и в связи с тем, что она часто возникает как подзадача в ряде разделов теории графов, которые затронуты в этой книге. В частности, большую роль она играет при вычислении хроматических чисел (гл. 4), нахождении центров графа (гл. 5) и паросочетаний (гл. 12).

[1]) Применяются также и другие названия: «*вершинное число независимости*» и «*число внутренней устойчивости*».— *Прим. ред.*

2. Независимые множества

Рассмотрим неориентированный граф $G = (X, \Gamma)$. *Независимое множество вершин* (известное также как *внутренне устойчивое множество* [11]) есть множество вершин графа G, такое, что любые две вершины в нем не смежны, т. е. никакая пара вершин не соединена ребром. Следовательно, любое множество $S \subset X$, которое удовлетворяет условию

$$S \cap \Gamma(S) = \varnothing, \tag{3.1}$$

является независимым множеством вершин. Например, для графа, приведенного на рис. 3.1, множества вершин $\{x_7, x_8, x_2\}$, $\{x_3, x_1\}$, $\{x_7, x_8, x_2, x_5\}$ — независимые. Когда не могут возникнуть недоразумения, эти множества будут называться просто независимыми множествами (вместо независимые множества вершин).

Независимое множество называется *максимальным*, когда нет *другого* независимого множества, в которое оно бы входило. Таким образом, множество S является *максимальным независимым множеством*, если оно удовлетворяет условию (3.1) и еще такому условию:

$$H \cap \Gamma(H) \neq \varnothing \qquad \forall H \supset S. \tag{3.2}$$

Следовательно, для графа, приведенного на рис. 3.1, множество $\{x_7, x_8, x_2, x_5\}$ является максимальным, а $\{x_7, x_8, x_2\}$ не является таковым. Множества $\{x_1, x_3, x_7\}$ и $\{x_4, x_6\}$ также являются максимальными независимыми множествами, и, значит, в данном графе больше одного независимого множества. Следует также отметить, что число элементов (вершин) в разных максимальных множествах, как следует из приведенного выше примера, не обязательно одинаковое.

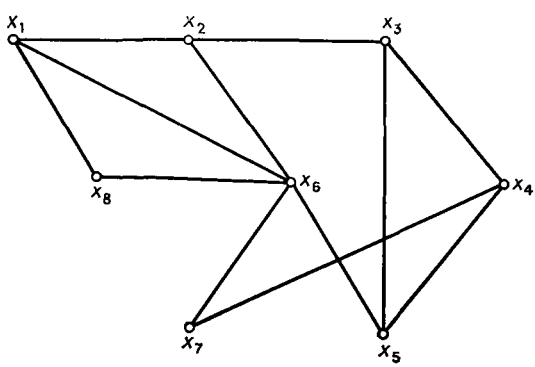


Рис. 3.1.

Если Q является семейством всех независимых множеств графа G, то число

$$\alpha[G] = \max_{S \in Q} |S| \tag{3.3}$$

называется *числом независимости* графа G, а множество S^*, на котором этот максимум достигается, называется *наибольшим независимым* множеством.

Для графа, приведенного на рис. 3.1, семейство максимальных независимых множеств таково:

$$\{x_8, x_7, x_2, x_5\}, \{x_1, x_3, x_7\}, \{x_2, x_4, x_8\}, \{x_6, x_4\},$$
$$\{x_6, x_3\}, \{x_7, x_5, x_1\}, \{x_1, x_4\}, \{x_3, x_7, x_8\}.$$

Наибольшее из этих множеств имеет 4 элемента и, следовательно, $\alpha[G] = 4$. Множество $\{x_8, x_7, x_2, x_5\}$ является наибольшим независимым множеством.

2.1. Пример: выбор проекта

Имеется n проектов, которые должны быть выполнены, и допустим, что для выполнения проекта x_i требуется некоторое подмножество R_i наличных ресурсов из множества $\{1, \ldots, p\}$. Далее предположим, что каждый проект, задаваемый совокупностью средств, необходимых для его реализации, может быть выполнен за один и тот же промежуток времени. Построим граф G, каждая вершина которого соответствует некоторому проекту, а ребро (x_i, x_j) — наличию общих средств обеспечения у проектов x_i и x_j, т. е. условию $R_i \cap R_j \neq \varnothing$. Максимальное независимое множество графа G представляет тогда максимальное множество проектов, которое можно выполнить одновременно за один и тот же промежуток времени.

В динамической системе происходит постоянное обновление проектов через определенный промежуток времени, так что каждый раз нужно заново повторять процедуру построения максимального независимого множества в соответствующем графе G. В практических ситуациях бывает весьма не просто выполнить множество проектов, соответствующих максимальному независимому множеству на данном отрезке времени, поскольку исполнение некоторых проектов может быть по каким-то причинам отложено. Тогда лучший способ действия состоит в присвоении каждому проекту (вершине) x_i некоторого штрафа p_i, который увеличивается с ростом времени отсрочки в исполнении проекта, и в каждый расчетный момент времени надо выбирать из семейства максимальных независимых множеств такое множество, которое максимизирует некоторую функцию штрафов на вершинах, содержащихся в выбранном множестве.

2.2. Максимальные полные подграфы (клики)

Понятие, противоположное максимальному независимому множеству, есть максимальный полный подграф. Таким образом, *максимальный полный подграф* (*клика*) графа G есть порожденный подграф, построенный на подмножестве S вершин графа и являющийся полным и максимальным в том смысле, что любой другой подграф графа G, построенный на множестве вершин H, содержащем S, т. е. $H \supset S$, не является полным. Следовательно, в противоположность максимальному независимому множеству, в котором не могут встретиться две смежные вершины, в клике все вершины попарно смежны. Совершенно очевидно, что максимальное независимое множество графа G соответствует клике графа $\widetilde{G}$ и наоборот, где $\widetilde{G}$ — дополнение графа G.

Вполне очевидно также, что понятие клики для неориентированного графа подобно понятию клики для графа; однако эта аналогия более глубокая, чем та, которая существует между понятиями клики и сильной компоненты (см. гл. 2). Клику в действительности можно рассматривать как такую сильную компоненту, в которой достижимость ограничена путями единичной длины (см. разд. 5 в гл. 2).

Аналогично тому, как было определено число независимости графа, с помощью соотношения (3.3) мы можем определить *кликовое число* графа (известное также как *густота* или *плотность* [1])). Это — максимальное число вершин в кликах данного графа. Тогда, образно говоря, у «плотного» графа кликовое число будет, вероятно, больше, а число независимости меньше, в то время как у «разреженного» графа, по всей вероятности, будет иметь место противоположное соотношение между этими числами.

2.3. Построение всех максимальных независимых множеств

Вследствие упомянутой выше связи между кликами и максимальными независимыми множествами методы, рассматриваемые в данном разделе и описываемые на «языке максимальных независимых множеств», могут быть непосредственно использованы для построения клик. На первый взгляд кажется, что нахождение всех максимальных независимых множеств графа — очень простая задача, которую можно решить последовательным перебором независимых множеств с одновременной проверкой каждого множества на максимальность (последнее осуществляется путем добавления к исследуемому множеству дополнительной, не принадлежащей ему, вершины и выяснения того, сохраняется ли

[1] В оригинале — «*density*».— *Прим. ред.*

независимость) и запоминанием максимальных множеств. Представление о простоте задачи действительно справедливо, но только для небольших графов, например с числом вершин, не превосходящим 20. Однако с увеличением числа вершин этот метод поиска становится с вычислительной точки зрения громоздким. Но все же громоздкость здесь не столь велика, как кажется с первого взгляда. Число максимальных независимых множеств увеличивается, но в процессе выполнения процедуры большое число независимых множеств формируется, а затем вычеркивается, так как обнаруживается, что они содержатся в других, ранее полученных множествах и поэтому не являются максимальными.

В этом разделе будет описан систематический метод перебора Брона и Кэрбоша [14], который позволяет обходить указанные выше трудности. В этом методе не нужно запоминать генерируемые независимые множества для проверки их на максимальность путем сравнения с ранее сформированными множествами.

2.3.1. Обоснование алгоритма. Этот алгоритм является существенно упрощенным перебором, использующим дерево поиска. В процессе поиска — на некотором этапе k — независимое множество вершин S_k расширяется путем добавления к нему подходящим образом выбранной вершины (чтобы получилось новое независимое множество S_{k+1}) на этапе $k+1$, и так поступают до тех пор, пока добавление вершин станет невозможным, а порождаемое множество не станет максимальным независимым множеством. Пусть Q_k будет на этапе k наибольшим множеством вершин, для которого $S_k \cap Q_k = \varnothing$, т. е. после добавления любой вершины из Q_k к S_k получается независимое множество S_{k+1}. В некоторый произвольный момент работы алгоритма множество Q_k состоит вообще говоря, из вершин двух типов: подмножества Q_k^- тех вершин, которые уже использовались в процессе поиска для расширения множеств S_k, и подмножества Q_k^+ таких вершин, которые еще не использовались. Тогда дальнейшее ветвление в дереве поиска включает процедуру выбора вершины $x_{i_k} \in Q_k^+$, добавление ее к S_k для построения множества

$$S_{k+1} = S_k \cup \{x_{i_k}\} \tag{3.4}$$

и порождение новых множеств:

$$Q_{k+1}^- = Q_k^- - \Gamma(x_{i_k}) \tag{3.5}$$

и

$$Q_{k+1}^+ = Q_k^+ - (\Gamma(x_{i_k}) \cup \{x_{i_k}\}). \tag{3.6}$$

Шаг возвращения алгоритма состоит в удалении вершины x_{i_k} из S_{k+1}, чтобы вернуться к S_k, изъятии x_{i_k} из старого множества

Q_k^+ и добавлении x_{i_k} к старому множеству Q_k^- для формирования новых множеств Q_k^+ и Q_k^-.

Легко заметить, что множество S_k является максимальным независимым множеством только тогда, когда невозможно его дальнейшее расширение, т. е. когда $Q_k^+ = \varnothing$. Если $Q_k^- \neq \varnothing$, то немедленно заключаем, что текущее множество S_k было расширено на некотором предшествующем этапе работы алгоритма путем добавления вершины из Q_k^-, и поэтому оно не является максимальным независимым множеством. Таким образом, необходимым и достаточным условием того, что S_k — максимальное независимое множество, является выполнение равенств

$$Q_k^+ = Q_k^- = \varnothing. \tag{3.7}$$

Теперь совершенно очевидно, что если очередной этап работы алгоритма наступает тогда, когда существует некоторая вершина $x \in Q_k^-$, для которой $\Gamma(x) \cap Q_k^+ = \varnothing$, то безразлично, какая из вершин, принадлежащих Q_k^+, используется для расширения S_k, и это справедливо при любом числе дальнейших ветвлений; вершина x не может быть удалена из Q_p^- на любом следующем шаге $p > k$. Таким образом, условие

$$\exists x \in Q_k^-, \text{ такая, что } \Gamma(x) \cap Q_k^+ = \varnothing, \tag{3.8}$$

является достаточным для осуществления шага возвращения, поскольку из S_k при всяком дальнейшем ветвлении уже не получится максимальное независимое множество.

Как и во всяком методе, использующем дерево поиска, здесь выгодно стремиться начать шаги возвращения как можно раньше, поскольку это ограничит «размеры» «ненужной» части дерева поиска. Следовательно, целесообразно сосредоточить усилия на том, чтобы возможно раньше добиться выполнения условия (3.8) с помощью подходящего выбора вершин, используемых при расширении множеств S_k. На каждом следующем шаге процедуры можно выбирать для добавления к S_k любую вершину $x_{i_k} \in Q_k^+$; на шаге возвращения x_{i_k} будет удалена из Q_k^+ и включена в Q_k^-. Если вершину x_k выбрать так, чтобы она принадлежала множеству $\Gamma(x)$ при некоторой вершине x из Q_k^-, то на соответствующем шаге возвращения величина

$$\Delta(x) = |\Gamma(x) \cap Q_k^+| \tag{3.9}$$

уменьшится на единицу (по сравнению с тем значением, которое было до выполнения прямого шага и шага возвращения), так что условие (3.8) теперь станет выполняться раньше.

Таким образом, один из возможных способов выбора вершины x_{i_k} для расширения множества S_k состоит, во-первых, в нахождении вершины $x^* \in Q_k^-$ с возможно меньшим значением величины

$\Delta(x^*)$ и, кроме того, в выборе вершины x_{i_k} из множества $\Gamma(x^*) \cap Q_k^+$. Такой выбор вершины x_{i_k} будет приводить на шаге возвращения к уменьшению величины $\Delta(x^*)$ — каждый раз на единицу — до тех пор, пока вершина x^* не станет удовлетворять условию (3.8) при выполнении шага возвращения.

Следует отметить, что поскольку на шаге возвращения вершина x_{i_k} попадает в Q_k^-, то может оказаться, что при этом новом входе значение величины Δ меньше, чем для ранее фиксированной вершины x^*. Значит, надо проверить, не ускорит ли эта новая вершина выполнение условия (3.8). Это особенно важно в начале ветвления, когда $Q_k^- = \varnothing$.

2.3.2. Описание алгоритма

Начальная установка

Шаг 1. Пусть $S_0 = Q_0^- = \varnothing$, $Q_0^+ = X$, $k = 0$.

Прямой шаг

Шаг 2. Выбрать вершину $x_{i_k} \in Q_k^+$, как упоминалось ранее, и сформировать S_{k+1}, Q_{k+1}^- и Q_{k+1}^+, оставляя Q_k^- и Q_k^+ нетронутыми. Положить $k = k + 1$.

Проверка

Шаг 3. Если удовлетворяется условие (3.8), то перейти к шагу 5, иначе к шагу 4.

Шаг 4. Если $Q_k^+ = Q_k^- = \varnothing$, то напечатать максимальное независимое множество S_k и перейти к шагу 5. Если $Q_k^+ = \varnothing$, а $Q_k^- \neq \varnothing$, то перейти к шагу 5. Иначе перейти к шагу 2.

Шаг возвращения

Шаг 5. Положить $k = k - 1$. Удалить x_{i_k} из S_{k+1}, чтобы получить S_k. Исправить Q_k^- и Q_k^+, удалив вершину x_{i_k} из Q_k^+ и добавив ее к Q_k^-. Если $k = 0$ и $Q_0^+ = \varnothing$, то остановиться. (К этому моменту будут уже напечатаны все максимальные независимые множества.) Иначе перейти к шагу 3.

Комментарий по применению описанного выше алгоритма приведен в [14] вместе с программой, написанной на Алголе. Эта программа была опробована для большого числа графов (в частности, для графов Муна—Мозера, см. задачу 10) и было установлено, что время, необходимое для построения максимального независимого множества, почти постоянно и не зависит от размера графа. Все это говорит о том, что данный алгоритм является одним из лучших.

3. Доминирующие множества

Для графа $G = (X, \Gamma)$ *доминирующее множество вершин* (называемое также *внешне устойчивым множеством* [11]) есть множество вершин $S \subseteq X$, выбранное так, что для каждой вершины x_j, не входящей в S, существует дуга, идущая из некоторой вершины множества S в вершину x_j.

Таким образом, S есть доминирующее множество вершин (или просто *доминирующее множество*, когда нет опасности возникновения путаницы), если

$$S \cup \Gamma(S) = X \tag{3.10}$$

Для графа, приведенного на рис. 3.2, множества вершин $\{x_1, x_4, x_6\}$, $\{x_1, x_4\}$, $\{x_3, x_5, x_6\}$ являются доминирующими множествами.

Доминирующее множество называется *минимальным*, если нет другого доминирующего множества, содержащегося в нем.

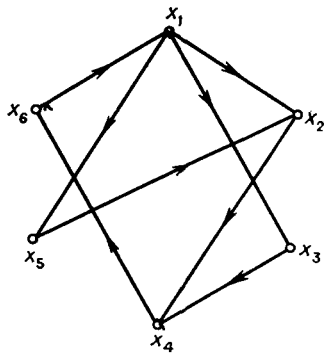


Рис. 3.2.

Таким образом, множество S является минимальным доминирующим множеством, если оно удовлетворяет соотношению (3.10) и нет собственного подмножества в S, которое удовлетворяет условию, аналогичному (3.10). Так, например, для графа, приведенного на рис. 3.2, множество $\{x_1, x_4\}$ — минимальное, а $\{x_1, x_4, x_6\}$ нет. Минимальным доминирующим множеством является также множество $\{x_3, x_5, x_6\}$, и еще существует несколько таких множеств в этом графе. Следовательно, как и в случае максимальных независимых множеств, в графе может быть несколько минимальных доминирующих множеств, и они не обязательно содержат одинаковое число вершин.

Если P — семейство всех минимальных доминирующих множеств графа, то число

$$\beta[G] = \min_{S \in P} |S| \tag{3.11}$$

называется *числом доминирования* [1]) графа G, а множество S^*, на котором достигается минимум, называется *наименьшим* доминирующим множеством.

Для графа, приведенного на рис. 3.2, наименьшим доминирующим множеством является множество $\{x_1, x_4\}$ и, следовательно, $\beta[G] = 2$.

3.1. Пример. Размещение «центров», покрывающих заданную область

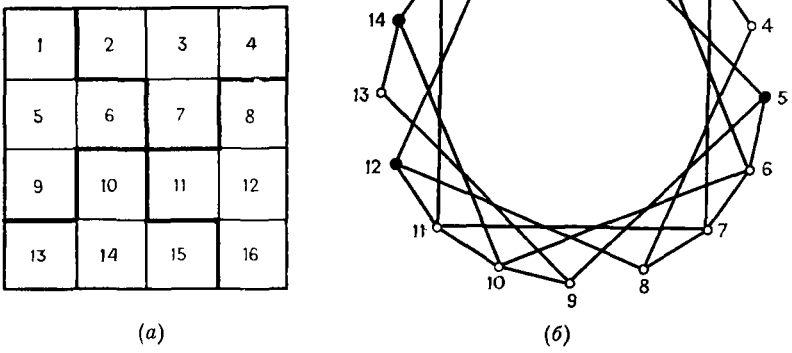
Задач такого типа весьма много. К ним относятся:

(а) Размещение телевизионных или радиопередающих станций на некоторой территории.

(б) Размещение военных баз, контролирующих данную территорию.

(в) Размещение центров торговли, обслуживающих некоторый район.

Предположим, что территория, представленная большим квадратом на рис. 3.3(а), разделена на 16 районов, как показано



(а) (б)

Рис. 3.3.

на рисунке. Предполагается, что военная база, расположенная в каком-либо районе, может контролировать не только этот район, но и соседние, граничащие с ним районы. Требуется найти наи-

[1]) В оригинале — «*dominance number*».— *Прим. ред.*

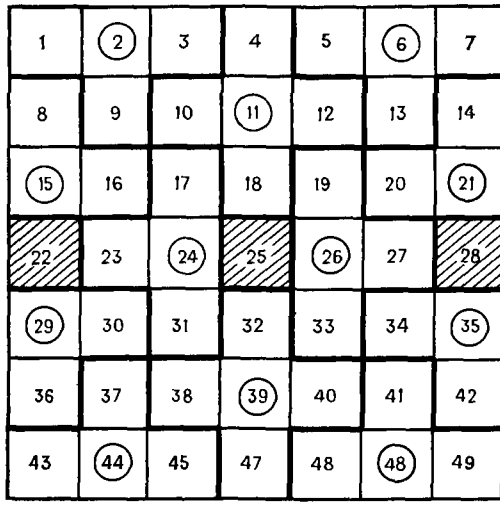


Рис. 3.4.

меньшее возможное число военных баз и места для их размещения, чтобы был обеспечен контроль всей территории.

Если мы представим каждый район вершиной графа и ребрами соединим только те пары вершин, которые соответствуют соседним районам, то получится граф, показанный на рис. 3.3(б). Тогда задача сводится к определению наименьшего доминирующего множества в этом графе. Число $\beta\,[G]$ является наименьшим числом баз, «покрывающих» всю территорию. Для графа, приведенного на рис. 3.3(б), $\beta\,[G] = 4$ и базы следует размещать в квадратах, номера которых принадлежат множеству {3, 5, 12, 14} или множеству {2, 9, 15, 8}.

Аналогично, для территории, показанной на рис. 3.4, число доминирования соответствующего графа равно 12 и базы следует размещать в районах 2, 6, 11, 15, 21, 24, 26, 29, 35, 39, 44, 48. Только три заштрихованных квадрата «защищены» одновременно тремя базами.

Связь между понятиями минимального доминирующего множества и базой графа почти очевидна, так же как между минимальным доминирующим множеством и p-центром (см. гл. 5). Эти взаимосвязи обсуждались в разд. 5 гл. 2, а в гл. 5 алгоритм нахождения наименьшего доминирующего множества используется в качестве основного шага более общего алгоритма построения p-центров.

Для случая максимальных независимых множеств мы привели алгоритм, который выдает полный список всех таких множеств.

Сделано это было потому, что такие списки нужны во многих практических задачах [36, 3, 2]. Для доминирующих множеств, однако, требуется обычно найти просто наименьшее доминирующее множество, и поэтому мы ограничимся здесь описанием алгоритма построения такого множества. В следующем разделе мы рассмотрим задачу о нахождении наименьшего доминирующего множества с несколько более общих позиций; это поможет нам глубже разобраться в взаимосвязях между понятиями, рассматриваемыми в других частях книги.

4. Задача о наименьшем покрытии

Пусть $\mathbf{A}^t$ — транспонированная матрица смежности графа G с единичными диагональными элементами. Задача определения наименьшего доминирующего множества графа G эквивалентна задаче нахождения такого наименьшего множества столбцов в матрице $\mathbf{A}^t$, что каждая строка матрицы содержит единицу хотя бы в одном из выбранных столбцов. Эта последняя задача о поиске наименьшего множества столбцов, «покрывающих» все строки, изучалась довольно интенсивно под названием *задачи о наименьшем покрытии* (ЗНП).

В общей ЗНП матрица, состоящая из 0 и 1, не обязательно является квадратной. Кроме того, каждому столбцу j (в нашем случае каждой вершине x_j) сопоставляется некоторая стоимость [1]) c_j и требуется выбрать покрытие (или, в другой терминологии — для случая графов — доминирующее множество вершин) с наименьшей общей стоимостью [2]). Поскольку задача построения наименьшего доминирующего множества вершин является весьма частной задачей о покрытии с $c_j = 1$ для всех $j = 1, \ldots, n$, то на первый взгляд кажется, что нахождение такого множества осуществляется на деле значительно проще, чем решение общей ЗНП. Однако это, вообще говоря, не так. Поэтому в данном разделе мы начнем с решения общей ЗНП.

4.1. Постановка задачи

ЗНП своим названием обязана следующей теоретико-множественной интерпретации. Даны множество $R = \{r_1, \ldots, r_M\}$ и семейство $\mathscr{S} = \{S_1, \ldots, S_N\}$ множеств $S_j \subset R$. Любое подсемейство $\mathscr{S}' = \{S_{j_1}, S_{j_2}, \ldots, S_{j_k}\}$ семейства $\mathscr{S}$, такое, что

$$\bigcup_{i=1}^{k} S_{j_i} = R, \tag{3.12}$$

[1] Или, в другой терминологии, «вес».— *Прим. ред.*

[2]) В другой терминологии —«покрытие наименьшего веса».— *Прим. ред.*

называется *покрытием множества* R, а множества S_{j_i} называются *покрывающими множествами*. Если в дополнение к соотношению (3.12) $\mathscr{S}'$ удовлетворяет условию

$$S_{j_h} \cap S_{j_l} = \varnothing, \quad \forall h, l \in \{1, \ldots, k\}, \quad h \neq l, \tag{3.13}$$

т. е. множества S_{j_i} $(i = 1, \ldots, k)$ попарно не пересекаются, то $\mathscr{S}'$ называется *разбиением множества* R.

Если каждому $S_j \in \mathscr{S}$ поставлена в соответствие (положительная) стоимость c_j, то ЗНП формулируется так: найти покрытие множества R, имеющее наименьшую стоимость, причем стоимость семейства $\mathscr{S}' = \{S_{j_1}, \ldots, S_{j_k}\}$ определяется как $\sum_{i=1}^{k} c_{j_i}$. Аналогично формулируется и *задача о наименьшем разбиении* (ЗНР).

В матричной форме, упомянутой ранее, когда строки $(M \times N)$-матрицы $[t_{ij}]$, состоящей из нулей и единиц, покрываются столбцами, ЗНП может быть сформулирована как задача линейного программирования:

$$\text{минимизировать } z = \sum_{j=1}^{N} c_j \xi_j$$

при ограничениях

$$\sum_{j=1}^{N} t_{ij} \xi_j \geqslant 1, \qquad i = 1, 2, \ldots, M, \tag{3.14}$$

где $c_j \geqslant 0$,

$$\xi_j = \begin{cases} 1, & \text{если } S_j \in \mathscr{S}', \\ 0, & \text{если } S_j \notin \mathscr{S}' \end{cases}$$

и

$$t_{ij} = \begin{cases} 1, & \text{если } r_i \in S_j, \\ 0, & \text{если } r_i \notin S_j. \end{cases}$$

Для ЗНР неравенства (3.14) обращаются в равенства

$$\sum_{j=1}^{N} t_{ij} \xi_j = 1, \qquad i = 1, 2, \ldots, M. \tag{3.15}$$

4.2. Упрощение задачи

Вследствие особой природы ЗНП часто удается сделать при ее исследовании определенные, хорошо известные заранее выводы и упрощения [6, 26, 27, 28, 30, 50, 51].

Например:

(1) если для некоторого элемента r_i из R справедливы соотношения $r_i \notin S_j$ $\forall j = 1, \ldots, N$, то r_i покрыть нельзя и, следовательно, задача не имеет решения;

(2) если $\exists r_i \in R$, такое, что $r_i \in S_k$ и $r_i \notin S_j$, $\forall j \neq k$, то S_k должно присутствовать во всех решениях и задачу можно свести к «меньшей», положив $R = R - \{r_i\}$ и $\mathscr{S} = \mathscr{S} - \{S_k\}$;

(3) пусть $V_i = \{j \mid r_i \in S_j\}$; тогда если $\exists p, q \in \{1, \ldots, M\}$ такие, что $V_p \subseteq V_q$, то r_q можно удалить из R, поскольку любое множество, которое покрывает r_p, должно также покрывать r_q, т. е. r_p доминирует над r_q;

(4) если для некоторого семейства множеств $\overline{\mathscr{S}} \subset \mathscr{S}$ справедливы соотношения $\bigcup_{S_j \in \overline{\mathscr{S}}} S_j \supseteq S_k$ и $\sum_{S_j \in \overline{\mathscr{S}}} c_j \leqslant c_k$ для любых $S_k \in \mathscr{S} - \overline{\mathscr{S}}$, то S_k может быть вычеркнуто из $\mathscr{S}$, поскольку $\bigcup_{S_j \in \overline{\mathscr{S}}} S_j$ доминирует над S_k.

Предположим, что все эти упрощения выполнены (если они возможны) и что исходная ЗНП уже переформулирована в соответствующей неприводимой форме.

4.3. Алгоритм решения ЗНР, использующий дерево поиска

Как отмечалось ранее, ЗНР тесно связана с ЗНП, являясь по существу ЗНП с дополнительным (неперекрываемость) ограничением. Это ограничение выгодно, если мы пытаемся решить задачу с помощью некоторого метода, использующего дерево поиска: при таком ограничении может рано выясниться, что некоторые возможные ветвления дерева рассматривать не надо. Учитывая вышесказанное, мы сначала займемся алгоритмом решения ЗНР (использующим дерево поиска), а затем покажем, как его можно приспособить к решению ЗНП.

Простые методы решения ЗНР, использующие дерево поиска, были предложены Пирсом [48] и Гарфинкелем и Немхаузером [27].

Сущность этих методов такова. Вначале строятся «блоки» столбцов, по одному на каждый элемент r_k из R, т. е. всего M блоков. k-й блок состоит из таких множеств семейства $\mathscr{S}$ (представленных столбцами), в которых содержится элемент r_k, но отсутствуют элементы с меньшими индексами — $r_1, \ldots, r_{k-1}$. Следовательно, каждое множество (столбец) появляется точно в одном определенном блоке и совокупность блоков может быть представлена в виде таблицы, как показано на рис. 3.1. В конкретных задачах некоторые из блоков могут отсутствовать.

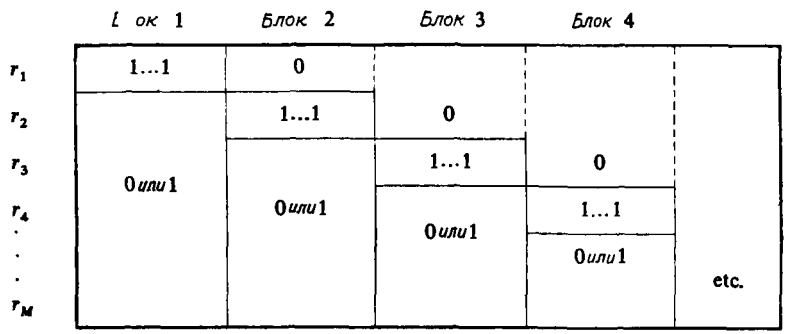
В процессе работы алгоритма блоки отыскиваются последовательно и формирование k-го блока начинается после того, как каждый элемент r_i, $1 \leqslant i \leqslant k-1$, будет покрыт частным решением. Таким образом, если какое-то множество в блоке k содержит элементы с индексами, меньшими k, то оно должно быть отбро-

шено (на этом этапе) в соответствии с требованием неперекрываемости.

Множества в пределах каждого блока размещаются в порядке возрастания их стоимостей и перенумеровываются так, что S_j теперь уже обозначает множество, соответствующее j-му столбцу таблицы.

Таблица 3.1

Исходная таблица



Текущее «наилучшее» решение $\check{B}$ со стоимостью $\check{z}$ известно на любом этапе поиска ($\check{B}$ обозначает семейство соответствующих покрывающих множеств). Если B и z — соответствующие семейство и стоимость на данной стадии поиска, а E — множество, представляющее те элементы (т. е. строки) r_i, которые покрываются множествами из B, то один из простых алгоритмов, использующих дерево поиска, можно описать следующим образом.

Присвоение начальных значений

Шаг 1. Построить исходную таблицу и начать с частного решения: $B = \varnothing$, $E = \varnothing$, $z = 0$ и $\check{z} = \infty$.

Расширение

Шаг 2. Найти $p = \min [i \mid r_i \notin E]$. Над блоком p поставить метку (над его первым множеством, которое, как следует из построения таблицы, имеет наименьшую стоимость).

Шаг 3. Начиная с отмеченной позиции в блоке p, перебирать его множества S_j^p, скажем, в порядке возрастания индекса j.

(i) Если найдено множество S_j^p, такое, что $S_j^p \cap E = \varnothing$ и $z + c_j^p < \check{z}$ (где c_j^p — стоимость множества S_j^p), то перейти к шагу 5.

(ii) В противном случае, т. е. если блок p исчерпан или выбрано множество S_j^p, такое, что $z + c_j^p \geqslant \check{z}$, перейти к шагу 4.

Шаг возвращения

Шаг 4. B не может привести к лучшему решению. Если $B = \varnothing$ (т. е. блок 1 исчерпан), то алгоритм заканчивает работу и оптимальным решением является $\check{B}$. В противном случае удалить последнее множество, скажем, S_h^l, добавить его в B, положить $p = l$, поставить метку над множеством S_{h+1}^l, удалить предшествующую метку в блоке l и перейти к шагу 3.

Проверка нового решения

Шаг 5. Обновить данные: $B = B \cup \{S_j^p\}$, $E = E \cup S_j^p$, $z = z + c_j^p$. Если найдено лучшее решение $E = R$, то положить $\check{B} = B$, $\check{z} = z$ и перейти к шагу 4. Иначе перейти к шагу 2.

Если поиск оканчивается с исчерпыванием блока 1 (см. выше шаг 4), то целесообразно переставить блоки в порядке возрастания числа столбцов (множеств) в каждом блоке. Это может быть осуществлено (перед построением исходной таблицы) перенумерацией элементов (строк) $r_1, \ldots, r_M$ в порядке увеличения числа множеств из S, содержащих соответствующие элементы.

Прежде чем показать, как алгоритм для ЗНР распространяется на ЗНП, кратко упомянем о некоторых других методах, применяемых при решении ЗНР. В [49] Пиро и Ласки дали ряд модификаций приведенного выше основного алгоритма, используя в качестве вспомогательного средства линейное программирование; в [45] Мишо описал другой алгоритм неявного перебора, который основан на задаче линейного программирования, соответствующей ЗНР с «блочной» структурой (рассмотренной выше и игравшей вспомогательную роль).

Предлагались также алгоритмы, включающие итерации симплексного типа, как прямые [4, 5], так и двойственные [58, 37]. В [37] Йенсен успешно применил метод динамического программирования к определенному типу ЗНР.

4.4. Алгоритм решения ЗНП, использующий дерево поиска

В алгоритме, описанном выше в разд. 4.3, единственным шагом, характерным для ЗНР, является шаг 3 (i). Если удалить в этом шаге требование «неперекрываемости» $S_j^p \cap E = \varnothing$, то алгоритм может быть использован для ЗНП. Однако в этом случае исходная таблица будет несколько отличаться от табл. 3.2. Итак, положим $S_j = \{r_{j_1}, r_{j_2}, r_{j_3}, \ldots\}$, где $j_1 < j_2 < j_3 < \ldots$. Теперь недостаточно включить S_j только в блок j_1, поскольку без требования

«неперекрываемости» нельзя исключить S_j из рассмотрения, скажем, в блоке j_2, если r_{j_1} уже покрыто частным решением. Следовательно, S_j должно входить в каждый блок $j_1, j_2, j_3, \ldots$. С другой стороны, поскольку элемент r_{j_α} из S_j согласно «последовательной природе» поиска покрывается перед ветвлением на множествах блока β ($\beta > j_\alpha$), то теперь возможно удалить все элементы x_{j_α} из S_j перед введением S_j в любой блок $\beta > j_\alpha$ без какого-либо влияния на результат решения задачи. Эта тривиальная операция удаления с вычислительной точки зрения оказывается очень выгодной, поскольку исходная таблица в ЗНП теперь может быть сокращена с помощью правил, указанных в разд. 4.2, до значительно меньших размеров, что возможно осуществить без удаления элементов x_{j_α} ($j_\alpha < \beta$) из множеств блока β [48].

Здесь следует отметить, что алгоритм из разд. 4.3 является примитивным поисковым алгоритмом, использующим дерево поиска, и лишен каких-либо тонкостей, хотя при тщательном программировании может быть сделан весьма эффективным для ЗНР [27]; это не так для ЗНП, даже при довольно скромных размерах. Далее мы рассмотрим некоторые важные условия и вычисление нижних границ, которые могут быть использованы для ограничения дерева поиска и улучшения эффективности основного алгоритма.

4.4.1. Некоторые важные условия. Прежде чем устанавливать общие результаты, мы проиллюстрируем эти важные условия на примере. Рассмотрим случай, когда блок 1 содержит (среди других) множества $S_1 = \{r_1, r_4, r_6\}$ и $S_2 = \{r_1, r_3, r_4\}$ со стоимостями 3 и 4 соответственно, а блок 2 содержит множества $S_3 = \{r_2, r_3, r_5\}$ и $S_4 = \{r_2, r_4, r_5\}$, каждое со стоимостью, равной 2.

В процессе выполнения алгоритма из разд. 4.3 на некотором этапе получим $B_a = \{S_1, S_3\}$, $E_a = \{r_1, r_2, \ldots, r_6\}$, $z_a = 5$; затем ветвление будет продолжаться до тех пор, пока либо мы не найдем решение, которое лучше, чем текущее $\check{B}$, либо не установим, что S_1 и S_3 не могут одновременно появиться в оптимальном решении.

Далее, через много шагов, мы достигнем такой ситуации, когда

$$B_b = \{S_2, S_3\}, \quad E_b = \{r_1, \ldots, r_5\}, \quad z_b = 6$$

Здесь становится ясным, что дальнейшее ветвление делать не нужно, поскольку $E_b \subseteq E_a$ и $z_b \geqslant z_a$. Подобная картина имеет место также при

$$B_c = \{S_2, S_4\}, \quad E_c = \{r_1, \ldots, r_5\}, \quad z_c = 6.$$

Таким образом, возможно, имеет смысл хранить для каждого значения $z = 1, 2, \ldots, \check{z}$ некоторый (быть может, неполный)

список максимальных множеств E, которые уже получены для данных z (где под максимальным понимается такое множество, которое не содержится в другом множестве из этого списка). Эти списки множеств E можно затем использовать для ограничения поиска путем элиминации тех ветвлений, которые позже оказываются бесполезными. Вообще непрактично хранить *все* максимальные множества E любого уровня стоимости. Если бы это было сделано, то полученный метод был бы похож на такой подход к задаче, который характерен для динамического программирования (или мог бы рассматриваться как полный древовидный поиск с приоритетом по ширине).

Пусть мы сохранили некоторый список $L(z_i)$ множеств E, которые были получены в процессе выполнения алгоритма на некотором уровне с суммарной стоимостью z_i. Предположим, что на данном этапе $E = E'$, $B = B'$, $z = z'$ и мы заняты исследованием блока k (где $k = \min\{i \mid r_i \notin E'\}$ — см. шаг 2 в алгоритме из разд. 4.3) и выбором множества S_j^k со стоимостью c_j^k для следующего ветвления. Если $z' + c_j^k < \tilde{z}$, то ветвление в рассматриваемом алгоритме с этого этапа продолжается дальше и

$$E = E' \cup S_j^k, \qquad B = B \cup \{S_j^k\}, \qquad z = z + c_j^k,$$

независимо от каких-либо других соображений.

Однако можно также гарантировать (на шаге 3), что перед продолжением ветвления

$$E' \cup S_j^k \not\subseteq E_l, \quad \forall E_l \in L(z_i) \quad \text{и при всех } z_i,$$
$$\text{для которых } z' < z_i \leqslant z' + c_j^k. \tag{3.16}$$

Если S_j^k не удовлетворяет приведенному выше условию, то оно отбрасывается и рассматривается следующее множество S_{j+1}^k блока k, и т. д. Если S_j^k удовлетворяет условию (3.16), то можно продолжать ветвление дальше с S_j^k так же, как и раньше, но с обновленным списком $L(z)$, полученным добавлением множества $E' \cup S_j^k$ в $L(z' + c_j^k)$.

Поскольку, как упоминалось ранее, невозможно практически хранить полные списки $L(z_i)$, то должны быть использованы некоторые эвристические критерии для определения размеров этих списков и способов их обновления в процессе поиска.

(А) *Размеры списка.* Лучше, очевидно, исключать множества S_j^k, которые могут оказаться ветвлениями дерева на его начальных уровнях, так как это может привести к отбрасыванию больших частей возможного дерева поиска. Таким образом, интуитивно ясно, что имеет смысл оставлять списки больших размеров $L(z_i)$, соответствующие меньшим z_i. Это подтверждается дополнительно

тем, что условие (3.16) выполняется с большей вероятностью тогда, когда множества T содержат лишь несколько элементов, что имеет место для малых z_i-уровней.

(Б) *Обновление списков.* На определенном z_i-уровне текущее E-множество (пусть это множество E') является с большей вероятностью подмножеством такого множества E'' (из списка $L(z_i)$), которое получено из той же общей части дерева поиска. Это справедливо потому, что E' и E'' имеют общую большую часть цепи от начального узла до соответствующих этим множествам узлов дерева. Итак, кажется более выгодным расположить списки $L(z_i)$ как стеки и использовать алгоритм FIFO «первым пришел—первым ушел», согласно которому в случае переполнения стека отбрасывается множество, находящееся внизу.

4.4.2. Вычисление нижней границы. На некотором этапе поиска, определяемом B', E', z', и когда блок k является следующим блоком, подлежащим рассмотрению, нижняя граница h для наименьшего значения величины z может быть вычислена и использована для ограничения дерева поиска следующим образом.

Рассмотрим непокрытый элемент $r_i \in R - E'$, который отсутствует в множествах блоков k, $k+1$, ..., $i-1$, соответствующих элементам, еще не покрытым частным решением. Тогда элемент r_i не может быть покрыт, пока некоторое множество S_j^i блока i не выбрано для добавления к B' на следующем этапе. Итак, для каждого такого элемента r_i строится строка для матрицы $D = [d_{qs}]$ и строка для второй матрицы $D' = [d'_{qs}]$, где d_{qs} равно числу элементов в множестве S_s^q, а d'_{qs} — стоимость множества S_s^q.

Кроме того, к каждой матрице D и D' добавляется дополнительная строка, скажем θ, с $d_{\theta s} = s$ для всех $s = 0, 1, \ldots, M - |E'|$ и $d'_{\theta s} = s \cdot \min [c_j^i / |S_j^i|]$, где минимумом берется по всем множествам S_j^i таким, что $r_i \notin E'$. Число элементов в строке q_1 матрицы D (или D') может быть отлично от числа элементов в другой строке q_2. Поэтому, добавив в конце строк 0 (нули) и ∞ (бесконечности) соответственно для матриц D и D', добьемся того, чтобы число элементов в разных строках стало одинаковым, скажем, равным f, а матрицы стали прямоугольными.

Теперь можно высказать ряд утверждений. Поскольку оптимальное решение текущей подзадачи должно покрывать $M - |E'|$ элементов, то, выбирая по одному значению из каждой строки матрицы D с таким расчетом, чтобы удовлетворялось условие

$$(d_{1s_1} + d_{2s_2} + \cdots + d_{\theta s_\theta}) \geqslant M - |E'|$$

и минимизировалась соответствующая стоимость

$$v = (d'_{1s_1} + \ldots + d'_{\theta s_\theta}),$$

мы как раз и получим нижнюю границу для оптимальной стоимости в подзадаче (о покрытии) — границей является найденное значение v. Здесь мы предполагали, что множества, соответствующие элементам матрицы D, расположенным в разных строках, не пересекаются; такая ситуация, очевидно, является наилучшей из возможных. Последняя строка θ просто гарантирует, что если

$$\sum_{i=1}^{\theta-1} d_{is_i} < M - |E'|,$$

то оставшиеся элементы покрываются наилучшим образом, т. е. с минимальной стоимостью покрытия для каждого дополнительно покрываемого элемента.

Наименьшее значение для

$$\sum_{i=1}^{\theta} d'_{is_i} \text{ при ограничении } \sum_{i=1}^{\theta} d_{is_i} \geqslant M - |E'|$$

легко может быть получено с помощью следующего алгоритма динамического программирования.

Пусть $g_\rho(v)$ — наибольшее число элементов, которые могут быть покрыты только с помощью первых ρ строк матрицы D (т. е. с использованием только ρ блоков задачи), причем общая стоимость покрытия не превышает v. Тогда $g_\rho(v)$ может быть найдено итерационным методом, так как

$$g_\rho(v) = \max_{s=1,\ldots,f} [d_{\rho s} + g_{\rho-1}(v - d'_{\rho s})], \tag{3.17}$$

где $g_0(v)$ придается начальное значение, равное 0 для всех v.

Следовательно, наименьшее из значений величины v, для которых выполняется неравенство $g_\theta(v) \geqslant M - |E'|$, как раз будет требуемой нижней границей h; оно может быть легко получено из таблицы решения задачи динамического программирования, составленной с использованием приведенного выше итерационного уравнения. Следует отметить, что необходимо рассмотреть только такие значения величины v, для которых $0 \leqslant v < \check{z} - z'$, поскольку если $h \geqslant z - z'$ (т. е. $g_\theta(v) < M - |E'|$ для $v \geqslant \check{z} - z'$), то можно сразу же сделать шаг возвращения.

4.5. Вычислительная характеристика алгоритма решения ЗНП

Вычислительные возможности алгоритма, описанного в разд. 4.4 и предназначенного для решения ЗНП, представлены в табл. 3.2. Во второй и третьей графах таблицы даны числа строк и столбцов, полученные после применения упрощающих

правил. Данные во всех задачах задавались случайным образом, а стоимости брались равными единице, т. е. $c_j = 1$, для каждого столбца j.

Таблица 3.2

Время вычисления при решении ЗНП

Задача	*Число строк*	*Число столбцов*	*Плотность*	*Время вычисления* [1]	*Узлы в дереве поиска* [2]
1	15	20	0,22	0,05	0,02
2	15	25	0,25	0,04	0,01
3	15	25	0,30	0,06	0,02
4	20	60	0,18	0,25	0,07
5	20	50	0,20	0,11	0,02
6	20	75	0,23	0,35	0,05
7	25	110	0,17	0,60	0,06
8	25	120	0,19	1,30	0,16
9	25	170	0,22	1,50	0,11
10	30	80	0,12	1,00	0,20
11	30	180	0,15	10,50	1,60
12	30	250	0,17	6,00	0,52
13	30	340	0,21	9,20	0,30
14	30	475	0,20	11,00	0,09
15	30	500	0,23	22,00	0,40
16	30	700	0,23	105,50	3,02
17	30	725	0,25	32,50	0,12
18	30	875	0,26	57,20	0,18
19	30	1000	0,27	72,80	0,19
20	35	160	0,09	18,50	2,18
21	35	290	0,13	28,00	2,03
22	35	460	0,16	75,50	2,95
23	35	585	0,18	129,50	3,96
24	35	780	0,19	141,20	5,03
25	35	1075	0,20	110,00	0,55

[1] СДС—6600, в секундах.
[2] Число узлов в дереве поиска (в тысячах).

Лемке, Салкин и Шпильберг [41] предложили такие методы решения ЗНП, в которых используются дерево поиска (иного типа), а также линейные программы. Решение соответствующей задачи линейного программирования берется в качестве нижней границы в процессе поиска и, кроме того, определяет характер последующего ветвления в текущем узле дерева поиска. Подходы, базирующиеся на рассмотрении отсекающих плоскостей и подобные, в принципе, тем, которые применяются в общем 0—1-программировании [32], представлены в работах Хауза, Нелсона и Радо [35] и Белмора и Рэтлифа [9]. Сравнение этих методов и исследование их вычислительных характеристик дается в статье Кристофидеса и Кормана [19].

5. Приложения задачи о покрытии

5.1. Выбор переводчиков

Предположим, что организации нужно нанять переводчиков с французского, немецкого, греческого, итальянского, испанского, русского и китайского языков на английский и что имеется пять кандидатур A, B, C, D и E. Каждая кандидатура владеет только некоторым собственным подмножеством из указанного выше множества языков и требует вполне определенную зарплату. Необходимо решить, каких переводчиков (с указанных выше языков на английский) надо нанять, чтобы затраты на зарплату были наименьшими. Очевидно, что это — задача о наименьшем покрытии.

Если, например, требования на оплату труда у всех претендентов одинаковые и группы языков, на которых они говорят, указаны ниже в матрице T, то решение задачи будет таким: нужно нанять переводчиков B, C и D.

Переводчик

Язык	*A*	*B**	*C**	*D**	*E*
Французский	1	0	1	1	0
Немецкий	1	1	0	0	0
Греческий	0	1	0	0	0
Итальянский	1	0	0	1	0
Испанский	0	0	1	0	0
Русский	0	1	1	0	1
Китайский	0	0	0	1	1

5.2. Информационный поиск [21]

Предположим, что некоторое количество единиц информации хранится в N массивах длины c_j, $j = 1, 2, \ldots, N$, причем на каждую единицу информации отводится по меньшей мере один массив. В некоторый момент делается запрос о M единицах информации. Они могут быть получены различными способами при помощи поиска в массиве. Для того чтобы получить все M единиц информации и при этом произвести просмотр массивов наименьшей длины, надо решить ЗНП, в которой элемент t_{ij} матрицы T равен 1, если информация i находится в массиве, и 0 в противном случае.

5.3. Маршруты полетов самолетов [1, 55, 64, 65]

Предположим, что вершины неориентированного графа G представляют аэропорты, а дуги графа G — этапы полетов (беспосадочные перелеты), которые осуществляются в заданное время. Любой маршрут в этом графе (удовлетворяющий ряду условий, которые могут встретиться на практике) соответствует некоторому реально выполнимому маршруту полета. Пусть имеется N таких возможных маршрутов и для каждого из них каким-то способом подсчитана его стоимость (например, стоимость j-го маршрута равна c_j). Задача нахождения множества маршрутов, имеющего наименьшую суммарную стоимость и такого, что каждый этап полета содержится хотя бы в одном выбранном маршруте, является задачей о наименьшем покрытии с матрицей $T = \|t_{ij}\|$, в которой элемент t_{ij} равен 1, если i-й этап содержится в j-м маршруте, и равен 0 в противном случае.

Укажем еще на одну разновидность рассмотренной задачи (также часто встречающуюся при назначении маршрутов полетов). Если требовать, чтобы каждый этап содержался только в одном маршруте, то приходим к ЗНР, соответствующей сформулированной выше ЗНП.

5.4. Упрощение логических (булевских) выражений [30, 50, 51, 67, 44, 43]

При упрощении логического выражения (логической формулы) E, которое, например, задано в дизъюнктивной нормальной форме [1]), достаточно рассмотреть только множество P простых импликантов формулы E. «Наипростейшая» форма[2]) для E получается тогда с помощью выделения в P подмножества $\check{P}$ наименьшей мощности, удовлетворяющего условию: каждая элементарная конъюнкция K из E «покрывается» по крайней мере одним простым импликантом J из $\check{P}$ (т. е. импликация $K \to J$ является тождественно истинной). Очевидно, что задача нахождения «наипростейшей» формы является ЗНП со стоимостями $c_j = 1$ для всех $j = 1, \ldots, N$.

Эта задача, кроме того, эквивалентна задаче построения «простейшей» переключательной (контактной) схемы, реализующей данную логическую формулу.

---

[1]) Часто применяют сокращение — д.н.ф.— *Прим. ред.*

[2]) «Наипростейшая» форма, рассматриваемая здесь, часто называется *кратчайшей д.н.ф.* Широко известны и другие д.н.ф.— *минимальные, тупиковые, сокращенные.— Прим. ред.*

5.5. Задача о развозке (о доставке) [7, 47, 24]

В задаче о развозке граф представляет сеть дорог, одна его вершина представляет склад, а все остальные вершины изображают потребителей. Транспорт, покидающий склад, снабжает товаром некоторых потребителей, после чего возвращается на склад. Пусть «стоимость» маршрута j равна c_j (например, c_j может быть километражем маршрута или временем, необходимым для его прохождения). Спрашивается, сколько машин следует использовать на разных маршрутах, чтобы в один и тот же день доставлять всем потребителям товары (каждому потребителю поставляется сразу все необходимое) и чтобы суммарная стоимость проходимых маршрутов была наименьшей. Эту задачу, очевидно, можно рассматривать как ЗНР, в которой столбцы представляют всевозможные осуществимые (с учетом практических ограничений) замкнутые маршруты, начинающиеся и кончающиеся на складе. Строки представляют потребителей.

Почти идентична рассмотренной задаче задача о прокладке электрического кабеля для подачи электроэнергии от подстанции (склада) к потребителям в «кольцевых схемах энергоснабжения» [15]. Другие практические приложения ЗНП и ЗНР связаны с синхронизацией линий сборки [59, 25], государственным районированием [29], с вычислением границ в задачах общего целочисленного программирования [18], с сетевым планированием [20] и с разработкой схем защиты и нападения [8, 10].

5.6. Другие задачи о покрытии графов

Большое число задач теории графов можно сформулировать как ЗНП, хотя многие из них могут быть решены более эффективно с помощью иных теоретико-графовых средств, рассмотренных в других частях этой книги. В данном разделе мы хотим показать, как некоторые из этих задач связаны с ЗНП.

5.6.1. Наименьшее доминирующее множество. Как упоминалось в разд. 4 настоящей главы, задача о нахождении наименьшего доминирующего множества графа G является ЗНП с такой матрицей T, которая получается в результате транспонирования матрицы смежности графа G с единичными элементами на главной диагонали.

5.6.2. Наибольшее независимое множество. Один из способов нахождения наибольшего независимого множества вершин графа $G = (X, \Gamma)$ состоит в построении всех максимальных независимых множеств вершин и выборе из них множества с наибольшей мощностью. Другой способ таков: исходная задача интерпретируется

как ЗНП, в которой столбцы матрицы T соответствуют вершинам графа G, а строки — ребрам, причем $t_{ij} = 1$, если вершина x_j инцидентна ребру a_i, и $t_{ij} = 0$ в противном случае.

Тогда очевидно [23], что если $\check{X} \subseteq X$ — множество столбцов в (наименьшем) решении рассматриваемой ЗНП, то множество $X - \check{X}$ является наибольшим независимым множеством графа G. Это справедливо потому, что если $\check{X}$ — множество, покрывающее ребра графа G, то каждое ребро инцидентно по крайней мере одной вершине из $\check{X}$ и, следовательно, никакие две вершины множества $X - \check{X}$ не могут быть смежными, т. е. $X - \check{X}$ — независимое множество. Более того, если $X - \check{X}$ является независимым, то каждое ребро графа инцидентно самое большее одной вершине из $X - \check{X}$ и, значит, $\check{X}$ — множество, покрывающее ребра графа G. Отсюда следует, что дополнение каждого множества вершин, покрывающего ребра графа G, является независимым множеством, а поскольку $\check{X}$ имеет наименьшую возможную мощность, то $X - \check{X}$ есть наибольшее независимое множество.

5.6.3. Наименьшее покрытие и наибольшее паросочетание (см. гл. 12). То, что в литературе называют *наименьшим «покрытием»*, представляет собой множество E ребер графа $G = (X, A)$, такое, что каждая вершина графа G инцидентна по крайней мере одному ребру из E и мощность множества E — минимально возможная. Таким образом, поскольку E можно рассматривать как «доминирующее над вершинами» графа G, то множество E^* — наименьшее из таких множеств — можно назвать, согласно терминологии, использованной в этой главе, *наименьшим доминирующим множеством ребер*. Известна иная задача о нахождении наименьшего покрытия: нужно отыскать «специальное» множество M ребер графа G — в M не должно быть смежных ребер. Множество M называют *паросочетанием*, а множество M^* — с наибольшей мощностью — является *наибольшим паросочетанием*, его можно называть также *наибольшим независимым множеством ребер*. Эквивалентность задач о наибольшем паросочетании и о наименьшем покрытии демонстрируется в гл. 12, посвященной изучению паросочетаний. Там устанавливается, в частности, следующее утверждение: пусть в наибольшем покрытии E^* степень вершины x_i есть $d^{E^*}(x_i)$ (рассматриваются только ребра из E^*), тогда, если для каждой вершины x_i с $d^{E^*}(x_i) > 1$ удалить $d^{E^*}(x_i) - 1$ ребер, инцидентных x_i, то оставшееся множество ребер образует наибольшее паросочетание. Обратно, если M^* есть наибольшее паросочетание и для каждой вершины x_i с с $d^{M^*}(x_i) = 0$ добавляется ребро, инцидентное x_i, то получающееся множество ребер образует наименьшее покрытие.

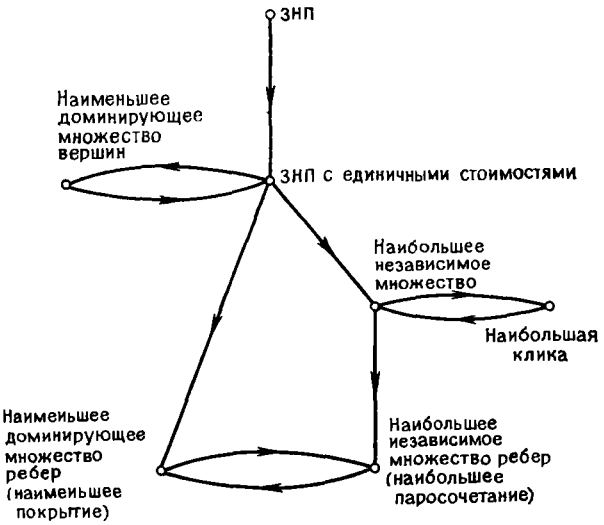


Рис. 3.5. Диаграмма взаимосвязей между задачами.

Наибольшие паросочетания и наименьшие покрытия можно описать на «языке» ЗНП. В случае покрытий столбцы матрицы T представляют ребра графа G, а строки — его вершины, причем $t_{ij} = 1$, если вершина x_i инцидентна ребру a_j, а иначе $t_{ij} = 0$. Следовательно, матрица T в этом случае есть матрица инциденций графа G.

На рис. 3.5 показана диаграмма взаимосвязей между задачами, где дуга от задачи α к эадаче β означает, что решение задачи α влечет за собой решение задачи β [34, 39]. Связь между наибольшими независимыми множествами вершин и наибольшими кликами устанавливается с помощью дополнительных графов, а между наибольшими независимыми множествами вершин и наибольшими паросочетаниями — с помощью реберных графов. (Определение реберного графа см. в гл. 10).

5.6.4. Покрытие графа подграфами. Известно целое семейство задач, связанных с покрытием (или разбиением) множества вершин или множества ребер графа специальными подграфами (на специальные подграфы). Например, можно рассматривать порожденные подграфы или остовные подграфы графа, имеющие предписанные свойства. Тогда в матрицах T из соответствующих ЗНП (или ЗНР) столбцы будут представлять все порожденные подграфы или остовные подграфы с заданными свойствами, а строки матриц будут представлять вершины или ребра графа. В гл. 4,

например, задача о нахождении хроматического числа графа G рассматривается как ЗНП, в которой строки матрицы T соответствуют вершинам графа G, а столбцы — максимальным независимым множествам вершин графа G (т. е. максимальным вполне несвязным подграфам графа G).

Другие задачи связаны с изучением покрытий ребер графа или его остовными подграфами с диаметром, не превосходящим λ [13], или «звездами» («звездными деревьями» графа), или простыми цепями и циклами [16, 42].

6. Задачи

1. Перечислить все максимальные независимые множества графа G, показанного на рис. 3.6, и, следовательно, найти число независимости $\alpha[G]$.

2. Используя метод из разд. 2.3.2, составить список всех максимальных независимых множеств графа, приведенного на рис. 3.7. (Обратить внимание на симметрию графа.)

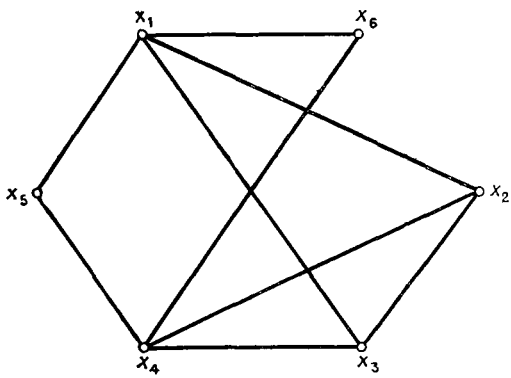


Рис. 3.6.

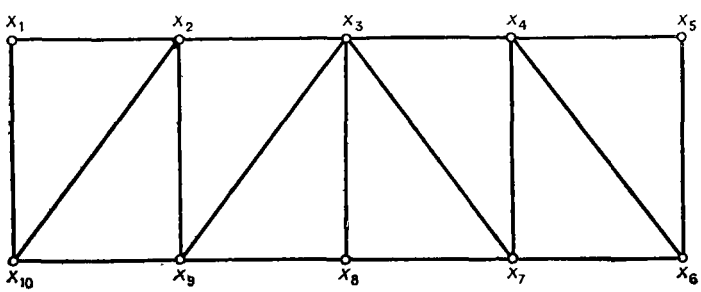


Рис. 3.7.

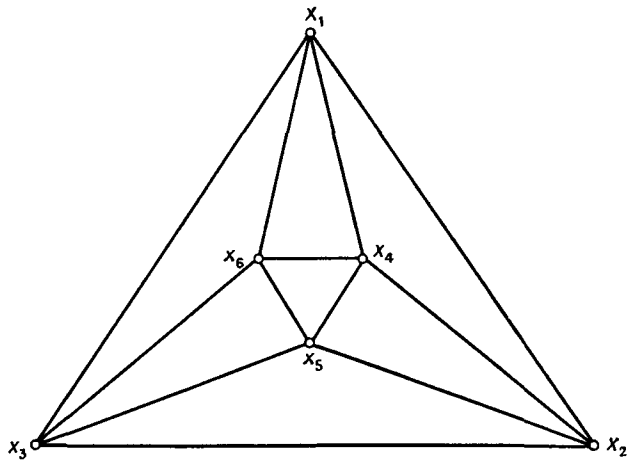


Рис. 3.8.

3. Показать, что для данного графа $G = (X, \Gamma)$ независимое множество $A \subset X$ является наибольшим тогда и только тогда, когда для произвольного независимого множества $B \subset X - A$ справедливо неравенство:

$$B \leqslant |\Gamma(B) \cup A|$$

(см. [63]).

4. Показать, что в полном неориентированном графе K_n каждое ребро принадлежит ровно $n - 2$ треугольникам (т. е. циклам длины 3).

5. Легко показать, что утверждение, обратное приведенному выше в задаче 4, неверно, т. е. если каждое ребро в графе G содержится ровно в $n - 2$ треугольниках, то граф G не обязательно совпадает с K_n. (На рис. 3.8 приведен граф, в котором каждое ребро принадлежит только двум треугольникам, но граф отличен от K_4.) Но приведенное в задаче 4 утверждение можно использовать для нахождения верхней оценки кликового числа графа G, т. е. такого наибольшего числа r, что в графе G содержится подграф K_r. Соответствующая оценка кликового числа графа G получается следующим образом:

(i) Начать со значения r, равного нижней границе кликового числа рассматриваемого графа.

(ii) Удалить те ребра, которые не содержатся хотя бы в $r - 2$ треугольниках.

(iii) Повторять (ii) для $r = r + 1$, $r + 2$ и т. д., до тех пор, пока не останется ребер меньше, чем $r(r - 1)/2$. Текущее зна-

чение r будет тогда верхней границей кликового числа. Показать, что описанный выше алгоритм является корректным, и применить его к графу $\widetilde{G}$ — дополнению графа G, приведенного на рис. 3.7. Результат сравнить со значением $\alpha[G]$, найденным в задаче 2.

6. Для передачи информации используется n символов (букв) из множества $X = \{x_1, \ldots, x_n\}$. Рассмотрим граф $G = (X, A)$, в котором $(x_i, x_j) \in A$ тогда и только тогда, когда символ x_i может быть спутан с символом x_j на приемном конце канала связи. Символы используются в блоках — k-буквенных «словах». Показать, что число, равное наибольшему числу слов, которые можно использовать для безошибочной передачи информации, совпадает с числом независимости $\alpha[G^k]$ графа $G^k = G \times G \times \ldots \times G$ (k раз), где произведение $G' \times G''$ двух графов $G' = (X', A')$ и $G'' = (X'', A'')$ есть граф $H = (Y, B)$, у которого

$$Y = \{x_i x_j \mid x_i \in X', x_j \in X''\}$$

и

$$B = \{(x_i x_j, x_k x_l) \mid x_i = x_k \text{ и } (x_j, x_l) \in A'', \text{ или } x_j = x_l \text{ и } (x_i, x_k) \in A'', \text{ или } (x_i, x_k) \in A' \text{ и } (x_j, x_l) \in A''\}$$

(см. [61]).

7. Показать, что для любых двух графов G' и G'' справедливо неравенство

$$\alpha[G' \times G''] \geqslant \alpha[G'] \cdot \alpha[G''].$$

8. (i) Пусть G — неориентированный граф. Обосновать неравенство $\alpha[G] \geqslant \beta[G]$, показав, что каждое максимальное независимое множество есть доминирующее множество.

(ii) Приведите пример, показывающий, что наименьшее доминирующее множество может не быть независимым.

9. Показать, что на шахматной доске $k \times k$ при $2 \leqslant k \leqslant 4$ невозможно расставить k ферзей так, чтобы они не атаковали друг друга. Найти решение для $k = 5$.

10. Мун и Мозер [46] доказали, что наибольшее число клик (обозначим его $f(n)$), которые могут встретиться в графе с n вершинами, дается соотношениями:

$$3^{n/3}, \text{ если } n = 0 \pmod 3,$$
$$4 \cdot 3^{(n-4)/3}, \text{ если } n = 1 \pmod 3,$$
$$2 \cdot 3^{(n-2)/3}, \text{ если } n = 2 \pmod 3.$$

Показать, что оценка $f(n)$ достигается только для следующих графов G:

а) Если $n = 0 \pmod 3$, то граф G состоит из $n/3$ компонент и каждая компонента является треугольником.

б) Если $n = 1 \pmod 3$, то либо граф G имеет $(n-4)/3+1$ компонент, причем одна из них — цикл C_4, а другие — треугольники, либо в графе G два «отдельных» ребра и $(n-4)/3$ «отдельных» треугольников.

в) Если $n = 2 \pmod 3$, то граф G состоит из одного отдельного ребра и $(n-2)/3$ отдельных треугольников.

Проверить приведенное выше утверждение для $n = 3, 4, 5$.

11. Используя упрощения, о которых шла речь в разд. 4.2, исключить максимально возможное число строк и столбцов из задачи о покрытии, у которой матрицы $[c_j]$ и $[t_{ij}]$ имеют следующий вид:

$$[t_{ij}] = \begin{array}{c|cccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 2 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 3 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 4 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 5 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 6 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}$$

$$[c_j] = [\,4,\ 7,\ 5,\ 8,\ 3,\ 2,\ 6,\ 5\,]$$

(Замечание. Упрощение 4 применять только для случая $|\bar{\mathcal{F}}| = 1$.)

12. Используя метод из разд. 4, решить ЗНП, полученную из задачи 11 после осуществления всех необходимых упрощений.

7. Список литературы

1. Arabeyre J. P., Fearnley J., Steiger F. C., Teather W. (1969), The airline crew scheduling problem: A survey, *Transp. Sci.*, 3, p. 140.
2. Augustson J. G., Minker J. (1970), An analysis of some graph-theoretical cluster techniques, *J. of ACM*, 17, p. 571.
3. Balas E. (1970), Project scheduling with resource constraints, Applications of mathematical programming techniques, Beale, Ed., English Universities Press, London.
4. Balas E., Padberg M. W. (1972a), On the set covering problem, *Ops. Res.*, 20, p. 1152.
5. Balas E., Padberg M. W. (1972), On the set covering problem. II: An algorithm, Management Sciences Research Report No 295, Carnegie—Mellon University.
6. Balinski M. (1965), Integer programming: methods, uses, computation, *Man. Sci.*, 12, p. 253.

7. Balinski M. L., Quandt R. E. (1964), On an integer program for a delivery problem, *Ops. Res.*, **12**, p. 300.
8. Bellmore M., Greenberg H., Jarvis J. (1970). Multi-commodity disconnecting sets, *Man. Sci.*, **16**, p. 427.
9. Bellmore M., Ratliff H. D. (1971), Set covering and involuntary bases, *Man. Sci.*, **18**, p. 194.
10. Bellmore M., Ratliff H. D. (1971), Optimal defense of multi-commodity networks, *Man. Sci.*, **18**, p. 174.
11. Берж К. (1962), Теория графов и ее применения, ИЛ, М.
12. Bonner R. E. (1964), On some clustering techniques, *IBM J. Res. and Dev.*, 8, p. 22.
13. Bosák J., Rosa A., Znám S. (1966), On decompositions of complete graphs into factors with given diameters, Int. Symp. on theory of graphs, Dunod, Paris, p. 37.
14. Bron C., Kerbosch J. (1973), Algorithm 457 — Finding all cliques of an undirected graph, *Comm. of ACM*, **16**, p. 575.
15. Burstall R. M. (1967), Tree searching methods with an application to a network design problem, Machine Intelligence, Colin and Michie, Eds., Vol. 1, Oliver and Boyd, London.
16. Басакер Р., Саати Т., Конечные графы и сети, М., Наука, 1974.
17. Chamberlin D. D. (1971), The single assignment approach to parallel processing, *Proc. of AFIPS Conf.*, 39, p. 263.
18. Christofides N. (1971), Zero-one programming using non-binary tree search, *The Computer J.*, **14**, p. 418.
19. Christofides N., Korman S. (in press), A computational survey of methods for the set covering problem, *Man. Sci.*
20. Crowston W. B. (1968), Decision network planning models, Ph. D. Thesis, Carnegie — Mellon University.
21. Day R. H. (1965), On optimal extracting from a multiple file data storage system: An application of integer programming, *Ops. Res.*, **13**, p. 482.
22. Desler J. F. (1969), Degree constrained subgraphs, covers, codes and k-graphs, Ph. D. Thesis, Northwestern University, Evaston, Illinois.
23. Edmonds J. (1962), Covers and packings in a family of sets, *Bulletin of American Mathematical Soc.*, **68**, p. 494.
24. Eilon S., Watson-Gandy C. D. T., Christofides N. (1971), Distribution management: Mathematical models and practical analysis, Griffin, London.
25. Freeman D. R., Jucker J. V. (1967), The line balancing problem, *J. of Industrial Engineering*, **18**, p. 361.
26. Garfinkel R. S. (1970), Set covering; a survey, presented at XVII TIMS Conf., London.
27. Garfinkel R. S., Nemhauser G. L. (1969), The set partitioning problem: set covering with equality constraints, *Ops. Res.*, **17**, p. 848.
28. Garfinkel R. S., Nemhauser G. L. (1972), Integer Programming, Wiley, New York.
29. Garfinkel R. S. (1968), Optimal political districting, Ph. D. Thesis, The John Hopkins University.
30. Gimpel J. F. (1965), A reduction technique for prime implicant tables, *IEEE Trans.*, **EC-14**, p. 535.
31. Glover F. (1971), A note on extreme point solutions and a paper by Lemke, Salkin and Spielberg, *Ops. Res.*, **19**, p. 1023.
32. Gomory R. (1963), An algorithm for integer solutions to linear programs, Recent Advances in Mathematical Programming, Graves and Wolfe, Eds., McGraw-Hill, New York.
33. Hakimi S. L., Frank H. (1969), Maximum internally stable sets of a graph, *J. of Math. Anal. and Appl.*, 25, p. 296.
34. Hakimi S. L. (1971), Steiners' problem in graphs and its implications, *Networks*, **1**, p. 112.

35. Hause R., Nelson L., Rado T. (1965), Computer studies of a certain class of linear integer programs, Recent Advances in Optimization Techniques, Lavi and Vogel, Eds., Wiley, New York.
36. Jardine N., Sibson R. (1971), Mathematical taxonomy, Wiley, London.
37. Jensen P. A. (1971), Optimal networks partitioning, *Ops. Res.*, 19, p. 916.
38. Капилина Р. И., Шнейдер Б. Н. (1970), Задача о внутренней устойчивости и методы ее решения с помощью ЭВМ, в сб. «Труды научной и технической конференции по итогам научных исследований за 1968 — 1969 гг., М., Издательство Московского Энергетического института.
39. Karp R. M. (1972), Reducibility of combinatorial problems, Complexity of computer computations, Miller and Thatcher, Eds., Plenum Press, New York, p. 85. [Русский перевод см. «Кибернетический сборник», Новая серия, 12, стр. 16—38.]
40. Lawler E. L. (1966), Covering problems: Duality relations and a new method of solution, *J. of SIAM (Appl. Math.)*, 14, p. 1115.
41. Lemke C. E., Salkin H. M., Spielberg K. (1971), Set covering by single branch enumeration with linear programming subproblems, *Ops. Res.*, 19, p. 998.
42. Lovsaz L. (1966), On covering of graphs, Int. Symp. on theory of graphs, Dunod, Paris, p. 231.
43. Mayoh B. H., (1967), Simplification of logical expressions, *J. of SIAM (Appl. Math.)*, 15, p. 898.
44. McCluskey E. J.Jr. (1956), Minimization of boolean functions, *Bell. Syst. Tech. J.*, 35, 1417.
45. Michadu P. (1972), Exact implicit enumeration method for solving the set partitioning problem, *IBM J. of Res. and Dev.*, 16, p. 573.
46. Moon J. W., Moser L. (1965), On cliques in graphs, *Israel J. of Mathematics*, p. 23.
47. Pierce J. F. (1967), On the truck dispatching problem — Part I, IBM Scientific Centre Technical Report 320—2018.
48. Pierce J. F. (1968), Application of combinatorial programming to a class of all-zero-one integer programming problems, *Man. Sci.*, 15, p. 191.
49. Pierce J. F., Lasky J. S. (1973), Improved combinatorial programming algorithms for a class of all-zero-one integer programming problems, *Man. Sci.*, 19, p. 528.
50. Pyne J. B., McCluskey E. J.Jr. (1961), An essay on prime implicant tables, *J. of SIAM (Appl. Math.)*, 9, p. 604.
51. Quine W. V. (1952), The problem of simplifying truth functions, *American Mathematical Monthly*, 59, p. 521.
52. Roth R. (1969), Computer solutions to minimum-cover problems, *Ops. Res.*, 17, p. 455.
53. Roy B. (1972), An algorithm for a general constrained set covering problem, Computing and Graph Theory, Read, Ed. Academic Press, New York.
54. Roy B. (1969, 1970), Algèbre moderne et théorie des graphes, Vol. 1 and Vol. 2, Dunod, Paris.
55. Rubin J. (1971), A technique for the solution of massive set covering problems, with application to airline crew scheduling, IBM Philadelphia Scientific Center Technical Rept., No 320-3004.
56. Rutman R. A. (1964), An algorithm for placement of inter-connected elements based on minimum wire length, *Proc. of AFIPS Conf.*, 20, p. 477.
57. Salkin H. M., Koncal R. (1970), A pseudo dual all-integer algorithm for the set covering problem, Dept. of O. R. Technical Memo, No 204, Case Western Reserve University.
58. Salkin H. M., Koncal R. (1971), A dual algorithm for the set covering problem, Dept. of O. R. Technical Memo, No 250, Case Western Reserve University.

59. Salveson M. E. (1955), The assembly line balancing problem, *J. of Industrial Engineering*, 6, p. 18.
60. Selby G. (1970), The use of topological methods in computer-aided circuit layout, Ph. D. Thesis, University of London.
61. Shannon C. E. (1956), The zero-error capacity of a noisy channel, *Trans. Inst. Elect. Eng.*, 3, p. 3. [Русский перевод см. Шеннон К., Работы по теории информации и кибернетике, М.,ИЛ, 1963.]
62. Шнейдер Б. Н. (1970), Приложение алгебры множеств к решению задач из теории графов.
63. Starobinets S. M. (1973), On an algorithm for finding the greatest internally stable sets of a graph, *Engineering Cibernetics* 73, English Translation by Plenum Press, p. 873.
64. Steiger F. (1965), Optimization of Swiss Air's crew scheduling by an integer linear programming model, Swissair Report O.R.SDK 3.3.911.
65. Steiger F., Neiderer M. (1968), Scheduling air crews by integer programming, presented at IFIP Congress 68, Edinburgh.
66. Steinberg L. (1961), The background wiring problem; a placement algorithm, *J. of SIAM (Review)*, 3, p. 37.
67. Thiriez H. (1971), The set covering problem — a group theoretic approach, *Rev. Française d'Informatique et de Recherche Opérationnelle*, **V—3**, p. 83.
68. Wells M. B., Application of a finite set covering theorem to the simplification of boolean function expressions, Information Processing 62, Popplewell, Ed., North Holland, Amsterdam.

Глава 4

РАСКРАСКИ

1. Введение

Разнообразные задачи, возникающие при планировании производства, составлении графиков осмотра, хранении и транспортировке товаров и т. д., могут быть представлены часто как задачи теории графов, тесно связанные с так называемой «задачей раскраски». Графы, рассматриваемые в этой главе, являются неориентированными и не имеют петель; если специально не оговаривается иное, то под словом «граф» понимается именно такой граф.

Граф G называют *r-хроматическим*, если его вершины *могут быть раскрашены* с использованием r цветов (красок) так, что не найдется двух смежных вершин одного цвета. Наименьшее число r, такое, что граф G является r-хроматическим, называется *хроматическим числом* графа G и обозначается $\gamma(G)$. Задача нахождения хроматического числа графа называется *задачей о раскраске* (или *задачей раскраски*) графа. Соответствующая этому числу раскраска вершин разбивает множество вершин графа на r подмножеств, каждое из которых содержит вершины одного цвета. Эти множества являются независимыми, поскольку в пределах одного множества нет двух смежных вершин.

Вообще говоря, хроматическое число графа (так же как числа независимости и доминирования, рассмотренные в предшествующей главе) нельзя найти, зная только числа вершин и ребер графа. Недостаточно также знать степень каждой вершины, чтобы вычислить хроматическое число графа. В этом можно убедиться, рассматривая графы, приведенные на рис. 4.1(а) и 4.1(б). Эти графы имеют по $n=12$ вершин, $m=16$ ребер и одинаковые распределения степеней вершин d_i. Однако хроматические числа данных графов равны 4 и 2 соответственно. При известных величинах n (число вершин), m (число ребер) и $d_1, \ldots, d_n$ (степени вершин графа) можно получить верхнюю и нижнюю оценки для хроматического числа графа. Этим оценкам посвящен следующий раздел.

Задача нахождения хроматического числа произвольного графа явилась предметом многих исследований в конце XIX и в текущем столетии. Сейчас по этому вопросу известно большое количество интересных результатов. В этой главе, однако, мы не пытаемся обсудить эти результаты или хотя бы дать их краткий обзор. Мы вводим только такие понятия, которые нужны для построения

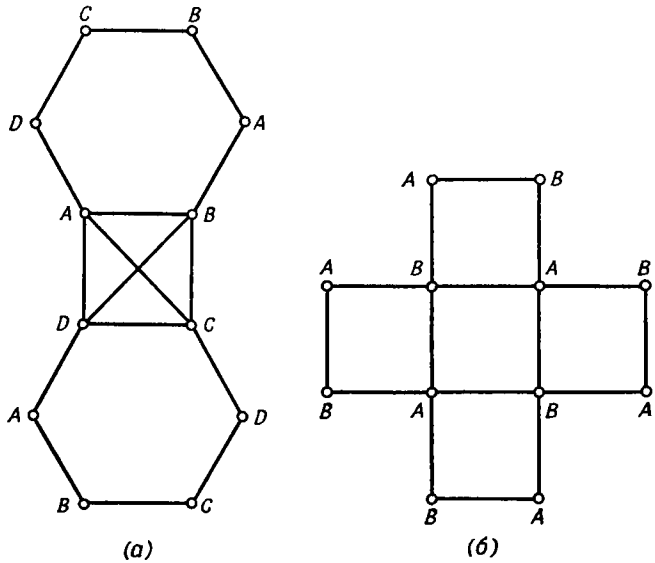


Рис. 4.1. Два графа с одинаковыми n, m и распределениями степеней вершин, но с различными хроматическими числами. (а) $\gamma(G) = 4$. (б) $\gamma(G) = 2$.

алгоритмов решения задачи о раскраске графа. Здесь мы рассматриваем в основном алгоритмы (как точные, так и «приближенные»), позволяющие находить (точное или приближенное) значение хроматического числа произвольного графа и соответствующую этому значению раскраску вершин.

2. Некоторые теоремы и оценки, относящиеся к хроматическим числам

В разд. 4 гл. 3 было введено понятие кликового числа $\rho(G)$ графа G (как наибольшего числа вершин в полном порожденном подграфе графа G) и было отмечено, что поскольку между кликами графа G и максимальными независимыми множествами дополнительного графа $\widetilde{G}$ существует взаимно однозначное соответствие, то справедливы равенства $\rho(G) = \alpha(\widetilde{G})$ и $\rho(\widetilde{G}) = \alpha(G)$.

2.1. Нижние оценки для $\gamma(G)$

Очевидно, поскольку по крайней мере $\rho(G)$ цветов требуется для раскраски соответствующей клики графа G (той самой клики, которая «определяет» кликовое число графа G), что $\rho(G)$ является

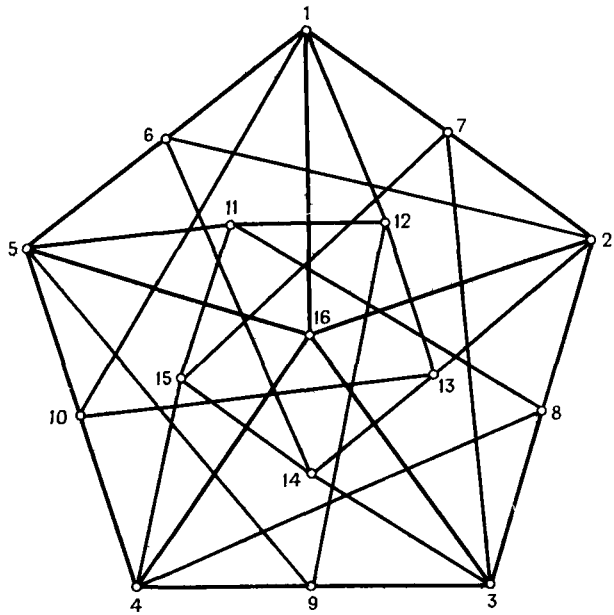


Рис. 4.2. Граф с $\rho(G) = 2$ и $\gamma(G) = 5$.

нижней оценкой хроматического числа, т. е.

$$\gamma(G) \geqslant \rho(G). \tag{4.1}$$

Более того, Зыков [26] доказал, что эта оценка — точная и что разность $\gamma(G) - \rho(G)$ может быть сколько угодно большой. В действительности Татт [22] показал, что можно построить граф G, который не содержит даже полного подграфа третьего порядка (т. е. $\rho(G) = 2$) и который будет иметь произвольно большое заданное значение хроматического числа. На рис. 4.2 изображен граф с $\rho(G) = 2$ и $\gamma(G) = 5$.

Поскольку число $\alpha(G)$ равно мощности наибольшего множества попарно несмежных вершин графа G, то оно совпадает также с мощностью наибольшего множества вершин в G, которые могут быть окрашены в один цвет, и, следовательно,

$$\gamma(G) \geqslant \left\lceil \frac{n}{\alpha(G)} \right\rceil, \tag{4.2}$$

где n — число вершин графа G, а $\lceil x \rceil$ обозначает наибольшее целое число, не превосходящее числа [1]) x.

[1]) Число $\lceil x \rceil$ называется «*целой частью числа x*» и часто обозначается через $[x]$. — *Прим. ред.*

Если через $\gamma(\bar{G})$ обозначить хроматическое число дополнения графа G, то можно записать следующие два неравенства, полученные Нордхаузом и Гаддумом [19]:

$$\gamma(G) \geqslant \lfloor 2\sqrt{n} \rfloor - \gamma(\bar{G}) \tag{4.3}$$

и

$$\gamma(G) \geqslant \frac{n}{\gamma(\bar{G})}, \tag{4.4}$$

где $\lfloor x \rfloor$ означает наименьшее целое число, которое не меньше [1]) x.

Еще одна нижняя оценка для $\gamma(G)$ предложена Геллером [8]:

$$\gamma(G) \geqslant \frac{n^2}{n^2 - 2m}. \tag{4.5}$$

Майерс и Лин [18] показали, что оценка из (4.1) равномерно мажорирует приведенную выше, и, следовательно, единственное преимущество оценки (4.5) состоит в том, что ее проще вычислять, чем оценку (4.1).

2.2. Верхние оценки для $\gamma(G)$

Нижние оценки хроматического числа безусловно более интересны, чем верхние, поскольку (если они достаточно близки к истинному значению) они могут быть использованы в процедуре вычисления $\gamma(G)$, включающей дерево поиска. В то же время верхние оценки хроматического числа подобного применения не находят. Тем не менее в литературе приводятся формулы для вычисления верхних оценок хроматического числа; так Брукс [3] предложена следующая легко вычисляемая оценка:

$$\gamma(G) \leqslant 1 + \max_{x_i \in X} [d(x_i) + 1]. \tag{4.6}$$

Другие достаточно точные верхние оценки, также использующие степени вершин графа, можно найти у Уэлша [23] и Вилфа и Секереша [21].

Верхние оценки, связывающие значения $\gamma(G)$ и $\gamma(\bar{G})$, приводятся у Нордхауза и Гаддума [19]:

$$\gamma(G) \leqslant n + 1 - \gamma(\bar{G}) \tag{4.7}$$

и

$$\gamma(G) \leqslant \left\lceil \left(\frac{n+1}{2}\right)^2 \right\rceil \Big/ \gamma(\bar{G}). \tag{4.8}$$

Оценки, даваемые формулами (4.3), (4.4), (4.7) и (4.8), являются наилучшими в том смысле, что можно построить графы, на которых эти оценки достигаются. В большинстве случаев, однако, они столь неточны, что не имеют никакой практической значимости.

[1]) Это число часто обозначается символом $]x[$.— *Прим. ред.*

2.3. Гипотеза четырех красок

Граф, который можно так изобразить на плоскости, что никакие два его ребра не пересекаются между собой [1]), называется *планарным*. Планарные графы важны как с теоретической, так и с практической точек зрения и обладают рядом таких свойств, связанных с раскраской, о которых следует упомянуть.

Теорема о пяти красках [13]

Каждый планарный граф можно так раскрасить, используя пять цветов, что любые две смежные вершины будут окрашены в разные цвета, т. е. если G — планарный граф, то $\gamma(G) \leqslant 5$.

«Теорема» о четырех красках (*недоказанная* [2])

Каждый планарный граф можно так раскрасить, используя 4 цвета, что любые две смежные вершины будут окрашены в разные цвета, т. е. $\gamma(G) \leqslant 4$, если G — планарный граф.

Впервые примерно в 1850 г. о гипотезе четырех красок речь шла в беседах Августа де Моргана и его ученика Ф. Гутри. Затем о ней говорилось в письме де Моргана сэру Вильяму Рауэну Гамильтону, датированном 23 октября 1852 г. Поскольку тогда довольно быстро нарастал поток «доказательств», «контрдоказательств», гипотез и теорем, относящихся к этой тематике, то накопилась громадная литература по гипотезе четырех красок, и эта «теорема» стала, по-видимому, самой знаменитой нерешенной задачей в математике. Мы не будем здесь подробно обсуждать гипотезу четырех красок и интересующегося ею читателя отсылаем к замечательной работе Оре [2].

3. Точные алгоритмы раскраски

В этом разделе рассматриваются некоторые алгоритмы, которые являются точными в том смысле, что они гарантируют нахождение оптимальной раскраски и истинного значения хроматического числа для любого графа.

---

[1]) Точнее, никакие две кривые, представляющие ребра графа, не пересекаются (геометрически) нигде, кроме инцидентной им обоим вершины (т. е. пересекаться могут только смежные ребра и притом лишь в «концевых точках».— *Прим. ред.*

[2]) Гипотезу четырех красок удалось обосновать с использованием ЭВМ (см. K. Appel, W. Haken, Every planar map is four colorable, *Bull. of Amer. Math. Soc.*, 82, № 5 (Sept. 1976)).— *Прим. ред.*

3.1. Метод динамического программирования

3.1.1. Максимальные r-подграфы. Порожденный подграф $\langle S_r[G]\rangle$ графа $G = (X, \Gamma)$, где $S_r[G] \subseteq X$, называется *r-подграфом*, если он r-хроматический. Если не существует такого множества H, что $H \supset S_r(G)$ и подграф $\langle H\rangle$ является r-хроматическим, то подграф $\langle S_r[G]\rangle$ называется *максимальным r-подграфом* графа G. Очевидно, что для фиксированного значения r, вообще говоря, существует много различных максимальных r-подграфов данного графа G. Подграф, у которого множество вершин совпадает с некоторым максимальным независимым множеством в G и который не имеет ребер (т. е. является вполне несвязным), есть максимальный 1-подграф графа G, поскольку в G нет 1-хроматического подграфа, имеющего большее множество вершин, чем у рассматриваемого подграфа.

Хроматическое число графа G можно определить как такое наименьшее число r, что $S_r[G] = X$ по крайней мере для одного из максимальных r-подграфов графа G.

Теорема 1. *Если граф G является r-хроматическим, то он может быть раскрашен с использованием r (или меньшего числа) красок с помощью следующей процедуры: сначала в один цвет окрашивается некоторое максимальное независимое множество $S_1[G]$, затем окрашивается в следующий цвет множество $S_1[\langle X - S_1[G]\rangle]$ и т. д. до тех пор, пока не будут раскрашены все вершины.*

Доказательство. Тот факт, что такая раскраска, использующая только r цветов, всегда существует, может быть установлен следующим образом. Пусть существует раскраска в r цветов, такая, что одно или больше множеств, окрашенных в один и тот же цвет, не являются максимальными независимыми множествами в смысле, упомянутом выше. Перенумеруем цвета произвольным способом. Очевидно, что мы можем всегда покрасить в цвет 1 те вершины (пусть это множество $\bar{V}_1$), которые не были окрашены в этот цвет и которые образуют максимальное независимое множество вместе с множеством V_1 всех вершин графа, уже окрашенных в цвет 1. Эта новая раскраска возможна потому, что никакая вершина из множества $\bar{V}_1$ не является смежной ни с какой вершиной из $\bar{V}_1$ и, следовательно, всякая вершина, которая смежна хотя бы с одной вершиной из $\bar{V}_1$, окрашена в цвет, отличный от цвета 1, и поэтому не затрагивается процедурой перемены цвета вершин из $\bar{V}_1$. Рассматривая теперь подграф $\langle X - V_1 \cup \bar{V}_1\rangle$ и проводя с ним аналогичные манипуляции, мы окрасим в цвет 2 какое-то (новое) максимальное независимое множество и т. д.

Раскраску указанного в теореме вида будем называть *оптимальной независимой раскраской.*

3.1.2. Рекуррентные соотношения. Вышеприведенную теорему можно использовать для получения рекуррентного соотношения, связывающего максимальные r-подграфы графа с его максимальными $(r-1)$-подграфами.

Пусть $Q_{r-1}[G]$ — семейство максимальных $(r-1)$-подграфов графа G, а $S^j_{r-1}[G]$ — множество вершин j-го подграфа из семейства $Q_{r-1}[G]$. Множество $S^h_1[G^j]$ является тогда множеством вершин k-го максимального 1-подграфа (независимого множества) графа $G^j \equiv \langle X - S^j_{r-1}[G]\rangle$, образованного вершинами графа G, не попавшими в $(r-1)$-подграф $\langle S^j_{r-1}[G]\rangle$.

Тогда следующим образом можно описать семейство всех максимальных r-подграфов графа G.

Строим множества H^i:

$$H^i = S^j_{r-1}[G] \cup S^h_1[G^j], \tag{4.9}$$

$j = 1, 2, \ldots, q_{r-1}$ и $k = 1, 2, \ldots, q^j_1$, где q_{r-1} и q^j_1 — соответственно число $(r-1)$-подграфов и число 1-подграфов.

Семейство максимальных r-подграфов содержится в семействе Θ множеств H^i $(i = 1, 2, \ldots, q^j_1 \cdot q_{r-1})$, и оно может быть получено из него исключением таких множеств семейства, которые содержатся в других множествах.

Итак, $\Theta_r[G]$ можно описать следующим образом:

$$\Theta_r[G] = \{H^i \mid H^i \in \Theta \quad \text{и} \quad H^i \not\subseteq H^j \quad \text{для любого } H^j \in \Theta,\ j \neq i\}. \tag{4.10}$$

3.1.3. Алгоритм, основанный на рассмотрении максимальных r-подграфов. В предыдущем разделе было указано, что хроматическое число графа G является таким наименьшим значением r, при котором $S_r[G]$ — множество вершин некоторого максимального r-подграфа — совпадает с X (множеством вершин графа G). Поэтому рекуррентные соотношения (4.9) и (4.10) могут быть использованы для последовательного построения максимальных 1-подграфов, 2-подграфов и т. д. и нахождения хроматического числа графа G, нужно просто на каждом шаге указанной процедуры проверять, не содержится ли множество вершин графа G в каком-нибудь из построенных подграфов [1]).

Ниже приводится описание алгоритма, позволяющего находить хроматическое число графа G и раскраску, соответствующую этому числу.

---

[1]) Алгоритм, описанный в этом разделе, можно толковать как алгоритм динамического программирования или как древовидный поиск (с приоритетом) по ширине. Недавно Вэном (*J. ACM*, 21, 1974, p. 385) был предложен иной, более выгодный, алгоритм, использующий древовидный поиск по глубине.

Шаг 1. Положить $r = 1$. Найти множества вершин $S_r^j[G]$, $j = 1, \ldots, q_r$, максимальных r-подграфов графа G. (Считаем, что имеется q_r таких множеств.) Пусть $Q = \{S_r^j[G] \mid j = 1, \ldots, q_r\}$. Положить $j = 1$.

Шаг 2. Найти максимальное независимое множество $S_1[G^j]$ графа $G^j = \langle X - S_r^j[G] \rangle$. Если такое множество существует, то перейти к шагу 3. Если все такие множества уже найдены, то перейти к шагу 6.

Шаг 3. Вычислить $S = S_r^j[G] \cup S_1[G^j]$.

Шаг 4. Если $S = X$, то остановиться. Число $r + 1$ есть хроматическое число графа G. Подмножества, включенные в множество S, дают требуемую раскраску. (Эти подмножества накапливаются по мере их введения и хранятся отдельно с соответствующими отметками (маркерами)). Если $S \neq X$, то перейти к шагу 5.

Шаг 5. (i) Если $S \subseteq S'$ для некоторого $S' \in Q$, то перейти к шагу 2.

(ii) Если $S \supset S'$ для некоторых $S' \in Q$, то положить $Q = Q - \{S'\}$ по всем таким S' из Q. Положить $Q = Q \cup \{S\}$ и перейти к шагу 2.

(iii) Если не выполняется ни одно из условий (i) и (ii), указанных выше, то положить $Q = Q \cup \{S\}$ и перейти к шагу 2.

Шаг 6. Если $j < q_r$, то положить $j = j + 1$ и перейти к шагу 2.

Если $j = q_r$, то положить $j = 1$, $r = r + 1$, $q_r =$ числу множеств в Q и перейти к шагу 2.

Если работу алгоритма не завершать при первом выполнении условия $S = X$ на шаге 4, то алгоритмический процесс будет продолжаться до получения раскраски в $r + 1$ цветов, если такая раскраска существует. Следует отметить, однако, что описанный алгоритм не дает полного перечисления всех возможных раскрасок в $r + 1$ цветов, а только порождает оптимальные независимые раскраски. Такие раскраски могут оказаться только небольшой частью всех возможных раскрасок в $r + 1$ цветов.

3.1.4. Пример. Рассмотрим семивершинный граф G, изображенный на рис. 4.3.

Шаг 1. Множествами вершин максимальных 1-подграфов являются: $S_1^1[G] = \{1, 4, 6\}$; $S_1^2[G] = \{2, 3, 5\}$; $S_1^3[G] = \{2, 5, 7\}$; $S_1^4[G] = \{2, 6\}$. Следовательно, $q_1 = 4$ и $Q = \{S_1^1[G], S_1^2[G], S_1^3[G], S_1^4[G]\}$.

Применяем (необходимое число раз) шаги 2—5 алгоритма:

Для $S_1^1[G]$

Шаг 2. $G^1 = \langle X - S_1^1[G] \rangle = \langle \{2, 3, 5, 7\} \rangle$; $S_1[G^1] = \{2, 3, 5\}$.

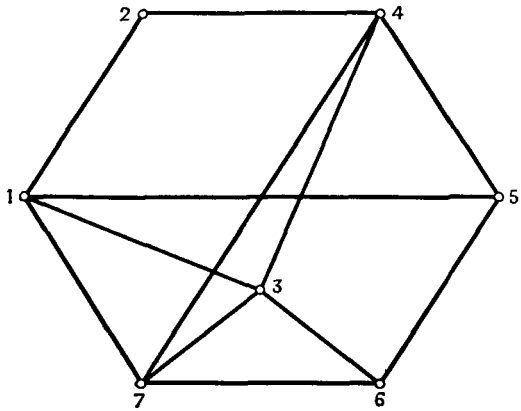


Рис. 4.3. Граф из примера 3.1.4.

Шаг 3. $S = \{1, 4, 6 \uparrow 2, 3, 5\}$.

Шаг 5. $Q = [\{1, 4, 6 \uparrow 2, 3, 5\}]$.

Шаг 2. $S_1[G^1] = \{2, 5, 7\}$.

Шаг 3. $S = \{1, 4, 6 \uparrow 2, 5, 7\}$.

Шаг 5. $Q = [\{1, 4, 6 \uparrow 2, 3, 5\}, \{1, 4, 6 \uparrow 2, 5, 7\}]$.

Для $S_1^2[G]$

Шаг 2. $G^2 = \langle\{1, 4, 6, 7\}\rangle$; $S_1[G^2] = \{1, 4, 6\}$.

Шаг 3. $S = \{2, 3, 5 \uparrow 1, 4, 6\}$, исключено в соответствии с шагом 5 (i).

Шаг 2. $S_1[G^2] = \{7\}$.

Шаг 3. $S = \{2, 3, 5 \uparrow 7\}$.

**Шаг 5.* $Q = [\{1, 4, 6 \uparrow 2, 3, 5\}, \{1, 4, 6 \uparrow 2, 5, 7\}, \{2, 3, 5 \uparrow 7\}]$.

Для $S_1^3[G]$

Шаг 2. $G^3 = \langle\{1, 3, 4, 6\}\rangle$; $S_1[G^3] = \{1, 4, 6\}$.

Шаг 3. $S = \{2, 5, 7 \uparrow 1, 4, 6\}$, исключено в соответствии с шагом 5(i).

Шаг 2. $S_1[G^3] = \{3\}$.

Шаг 3. $S = \{2, 5, 7 \uparrow 3\}$, исключено в соответствии с шагом 5(i).

Для $S_1^4[G]$

Шаг 2. $G^4 = \{1, 3, 4, 5, 7\}$, $S_1[G^4] = \{1, 4\}$.

Шаг 3. $S = \{2, 6 \uparrow 1, 4\}$, исключено в соответствии с шагом 5(i).

Шаг 2. $S_1[G^4] = \{3, 5\}$.

Шаг 3. $S = \{2, 6 \uparrow 3, 5\}$, исключено в соответствии с шагом 5(i).

Шаг 2. $S_1[G^4] = \{5, 7\}$.

Шаг 3. $S = \{2, 6 \uparrow 5, 7\}$, исключено в соответствии с шагом 5 (i).

Таким образом, в конце первой итерации, использующей шаги 2—5 алгоритма, получается семейство Q множеств $S_2^j[G]$, соответствующее максимальным 2-подграфам и показанное на шаге, отмеченном звездочкой. Следует обратить внимание на то, что 5 из восьми полученных множеств исключены в результате применения первого правила из шага 5 алгоритма.

Действуя аналогичным образом дальше, можно построить максимальные 3-подграфы и т. д. В действительности уже следующее порожденное множество $S_3^1[G] = \{1, 4, 6 \uparrow 2, 5, 3 \uparrow 7\}$ будет удовлетворять условию $S_3^1[G] = X$ на шаге 4. Следовательно, хроматическое число графа равно 3 и оптимальная раскраска задается следующим разбиением (множества X): $\{1, 4, 6\}$, $\{2, 5, 3\}$, $\{7\}$. Если применять алгоритм дальше, то будет получена еще одна возможная раскраска:

$$S_3^2[G] = \{1, 4, 6 \uparrow 2, 5, 7 \uparrow 3\} = X.$$

Все другие множества $S_3^j[G]$ либо содержатся в двух предшествующих множествах, либо не содержат все вершины из X.

Заметим, что такие раскраски, как $\{5, 3\}$, $\{1, 4, 6\}$, $\{2, 7\}$, хотя и возможны, но не являются оптимальными независимыми и не порождаются описанным выше алгоритмом.

3.2. Формулировка задачи о раскраске на языке 0-1-программирования

Пусть q — какая-нибудь верхняя оценка хроматического числа графа G. Эта верхняя оценка может быть одной из оценок, данных в разд. 2 этой главы, или может быть равна числу цветов, получающемуся в одном из тех эвристических методов решения задачи о раскраске, которые подробно будут описаны позже.

Пусть $\Xi = [\xi_{ij}]$ — матрица раскраски графа (задающая некоторую конкретную раскраску [1]) вершин графа), так что

$$\xi_{ij} = \begin{cases} 1, \text{ если вершина } x_i \text{ окрашена в цвет } j, \\ 0 \text{ в противном случае.} \end{cases}$$

[1]) Здесь не предполагается, что смежные вершины должны быть окрашены в разные цвета. Если же это условие выполняется, то раскраска называется *допустимой.— Прим. ред.*

Если $A = [a_{ik}]$ — матрица смежности графа G с диагональными элементами, равными 0, то следующие два условия гарантируют допустимость раскраски вершин графа G (т. е. допустимость матрицы раскраски $[\xi_{ij}]$):

$$\sum_{j=1}^{q} \xi_{ij} = 1 \text{ (для всех } i = 1, \ldots, n), \tag{4.11}$$

$$L \cdot (1 - \xi_{ij}) - \sum_{k=1}^{n} a_{ik}\xi_{kj} \geqslant 0 \quad \begin{cases} \text{для всех } i = 1, \ldots, n \\ \text{и} \qquad\quad j = 1, 2, \ldots, q. \end{cases} \tag{4.12}$$

Условие (4.11) обеспечивает раскраску вершины в один и только один цвет.

В условии (4.12) L — очень большое положительное число (любое целое, большее чем n). Если вершина x_i окрашена в цвет j (т. е. $\xi_{ij} = 1$), то первый член в (4.12) равен 0. Тогда и второй член должен быть равен 0, чтобы выполнялось неравенство, поскольку числа a_{ik} и ξ_{kj} неотрицательны. Таким образом, условие (4.12) обеспечивает допустимость раскраски, т. е. если вершина x_i окрашена в цвет j, то нет смежной с x_i вершины того же цвета. Если вершина x_i окрашена в цвет, отличный от j ($\xi_{ij} = 0$), то первый член в (4.12) равен L. Поскольку второй член в (4.12) не может, очевидно, достигнуть значения L (его наибольшее значение равно в действительности $n - 1$), то какое бы число вершин x_k, смежных с вершиной x_i, ни было окрашено в цвет j, неравенство (4.12) по-прежнему будет выполняться. Заметим, что если вершины x'_k и x''_k смежны с x_i, а также смежны между собой, то условие (4.12), записанное для вершины x_i, не будет препятствовать раскраске x'_k и x''_k в один и тот же цвет j. Однако, записав условие (4.12) для x'_k (или x''_k), мы обеспечим тем самым раскраску этих двух вершин (x'_k и x''_k) в разные цвета.

Пусть теперь каждому цвету j сопоставлен штраф p_j, выбранный так, что

$$p_{j+1} > h \cdot p_j \text{ (штраф } p_1 \text{ принят равным единице)} \tag{4.13}$$

и где h есть верхняя оценка для наибольшего числа вершин в графе, которые могут быть окрашены в один цвет, т. е. h — произвольное число, большее чем $\alpha(G)$ — число независимости графа. При отсутствии лучшей оценки можно положить $h = n$, не проводя лишних вычислений.

Задача раскраски вершин графа с использованием наименьшего числа цветов может быть сформулирована следующим образом:

минимизировать

$$z = \sum_{j=1}^{q} \sum_{i=1}^{n} p_j \xi_{ij} \tag{4.14}$$

при ограничениях (4.11) и (4.12).

Минимизация выражения (4.14) обеспечивает выполнимость следующего условия: цвет $j+1$ не будет использован в раскраске вершин, если цвета от 1 до j достаточны для допустимой раскраски. Матрица (раскраски графа) $[\xi^*_{ij}]$, которая дает решение приведенной выше задачи линейного 0-1-программирования, определяет оптимальную раскраску, а используемое при этом число цветов равно хроматическому числу графа.

Берж [1] вместо условия (4.12) предложил следующее:

$$\sum_{i=1}^{n} b_{ih}\xi_{ij} \leqslant 1 \quad \begin{cases} \text{для всех } k=1, 2, \ldots, m \\ \text{и} \qquad\quad j=1, 2, \ldots, q, \end{cases} \tag{4.15}$$

где $[b_{ik}]$ — матрица инциденций, т. е. $b_{ik}=1$, если вершина x_i инцидентна ребру a_k, и $b_{ik}=0$ в противном случае. Условие (4.15) отражает тот факт, что не более чем одна из двух концевых вершин любого ребра может быть окрашена в цвет j.

Хотя это условие более естественное, чем (4.12), но для его описания требуются mq ограничений, тогда как для условия (4.12) нужно только nq ограничений. Поскольку число ребер (m) связного графа обычно значительно больше числа его вершин (n), то условие (4.12) с вычислительной точки зрения предпочтительнее. Насколько велик при этом выигрыш, можно увидеть на следующем примере: если в некотором 100-вершинном графе число ребер будет составлять только 20% от числа ребер в полном 100-вершинном графе, то для задания условия (4.15) потребуется 1000 ограничений на каждый цвет, а для описания условия (4.12) — только 100 ограничений на каждый цвет.

3.3. Сведение задачи о раскраске к ЗНП

Поскольку при любой допустимой раскраске графа G множество вершин, окрашиваемых в один и тот же цвет, должно быть независимым множеством, то всякую допустимую раскраску можно интерпретировать как *разбиение* всех вершин графа G на такие независимые множества. Далее, если каждое независимое множество расширить до максимального (путем добавления к нему других вершин), то раскраска графа G может быть тогда истолкована как *покрытие* вершин графа G максимальными независимыми множествами. Очевидно, что в последнем случае некоторые вершины графа G могут принадлежать более чем одному максимальному независимому множеству. Это говорит о возможности существования различных допустимых раскрасок (использующих одно и то же число цветов), так как вершину, принадлежащую разным максимальным множествам, можно окрасить в любой из цветов, соответствующих тем максимальным независимым множествам, которым она принадлежит.

Итак, пусть построены все максимальные независимые множества графа G, например с помощью алгоритма, приведенного в разд. 2.3 предшествующей главы (пусть таких множеств t), и пусть задана $(n \times t)$-матрица $\mathbf{M} = [m_{ij}]$, у которой $m_{ij} = 1$, если вершина x_i принадлежит j-му максимальному независимому множеству, и $m_{ij} = 0$ в противном случае. Если теперь каждому максимальному независимому множеству сопоставить единичную стоимость, то задача раскраски сведется просто к задаче нахождения наименьшего числа столбцов в матрице $\mathbf{M}$, покрывающих все ее строки [1]). Каждый столбец из решения этой ЗНП соответствует определенному цвету, который может быть использован для окраски всех вершин максимального независимого множества, представленного этим столбцом.

Хотя t — число столбцов матрицы $\mathbf{M}$ — может быть весьма велико даже для графов средних размеров, формулировка задачи раскраски как ЗНП предпочтительнее, чем непосредственная ее постановка как задачи общего 0-1-программирования, потому что ЗПН могут быть решены для таких размеров (т. е. для числа переменных), которые по крайней мере на порядок больше, чем «подходящие» размеры в случае задач общего 0-1-программирования (см. гл. 3).

3.3.1. Пример. Для графа, изображенного на рис. 4.4, число всех максимальных независимых множеств равно 15, и в матрице, приведенной ниже, они представлены столбцами; на пустых местах в матрице должны стоять нули.

Максимальные независимые множества

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	x_1	1	1	1	1											
	x_2					1	1	1	1							
	x_3	1	1							1	1	1	1			
	x_4			1		1	1							1		
$\mathbf{M} =$	x_5	1			1			1		1						
	x_6					1			1		1	1		1	1	1
	x_7									1	1		1	1	1	
	x_8				1											1
	x_9		1						1			1	1		1	
	x_{10}						1									
			↑		↑		↑				↑					

[1]) Напоминаем, что i-й столбец в 0-1 *-матрице* покрывает те и только те строки, в которых на пересечении с i-м столбцом стоят единицы.— *Прим. ред.*

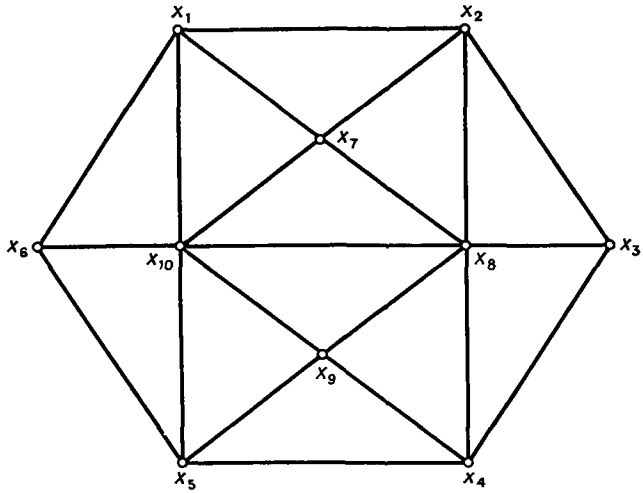


Рис. 4.4. Граф из примера 3.3.1.

Решение задачи о покрытии для этой матрицы состоит из 4 столбцов и не является единственным. Например, решениями будут множества столбцов {4, 6, 10, 14}, {4, 6, 10, 12}, {4, 6, 10, 11}, {4, 6, 10, 8} и {4, 6, 10, 2}. Если мы выберем последнее из этих решений (отмеченное стрелками) и сопоставим цвета a, b, c, d столбцам 4, 6, 10 и 2 соответственно, то вершину x_1 (принадлежащую одновременно столбцам 2 и 4) можно окрасить в цвет a или в цвет d, а вершину x_3 (принадлежащую обоим столбцам 2 и 10) можно окрасить в цвет c или в цвет d и таким образом получить различные оптимальные раскраски.

Интересно отметить, что в данном примере нижняя оценка для $\gamma\,[G]$ равна 4 (это можно получить из соотношения (4.2), так как $\alpha\,[G] = 3$ и $n = 10$) и, значит, совпадает с истинным значением $\gamma\,[G]$.

3.4. Алгоритм прямого неявного перебора, использующий дерево поиска

Для определения хроматического числа графа может быть использован с поразительной эффективностью очень простой метод неявного перебора, не содержащий никаких хитростей [4, 9]. Метод состоит в следующем.

Предположим, что множество вершин как-то упорядочено и x_i — i-я вершина этого множества. Тогда первоначальная допустимая раскраска может быть получена так:

(i) окрасить x_i в цвет 1.

(ii) каждую из оставшихся вершин окрашивать последовательно [1]): вершина x_i окрашивается в цвет с наименьшим возможным «номером» (т. е. выбираемый цвет должен быть первым в данном упорядочении цветом, не использованным при окраске какой-либо вершины, смежной с x_i).

Пусть q — число цветов, требуемое для вышеупомянутой раскраски. Если существует раскраска, использующая только $q-1$ цветов, то все вершины, окрашенные в цвет q, должны быть перекрашены в цвет $j<q$. Пусть x_{i*} — первая вершина при заданном упорядочении, которая была окрашена в цвет q. Поскольку (согласно (ii)) она была так окрашена потому, что не могла быть окрашена в цвет $j<q$, то ее можно перекрасить в цвет $j<q$, лишь перекрасив предварительно хотя бы одну из смежных с ней вершин. Итак, шаг возвращения из вершины x_{i*} можно осуществить следующим образом.

Из смежных с x_{i*} вершин в множестве $\{x_1, \ldots, x_{i*-1}\}$ найти последнюю (при заданном упорядочении), т. е. вершину с наибольшим индексом. Пусть это будет вершина x_k. Если x_k окрашена в цвет j_k, то x_k перекрашивается в другой допустимый цвет с наименьшим возможным «номером» j'_k, таким, что $j'_k \geqslant j_k+1$.

Если $j'_k<q$, то надо продолжать последовательно перекрашивать все вершины с x_{k+1} до x_n, применяя указанное выше правило (ii) и помня о том, что цвет q использовать нельзя. Если такая процедура осуществима, то будет найдена новая лучшая раскраска, использующая меньше, чем q цветов. В противном случае, т. е. если встретится вершина, «требующая» цвет q, то можно снова сделать шаг возвращения — из такой вершины.

Если $j'_k=q$ или нет другого допустимого цвета j'_k (см. ниже замечание (а)), то можно сразу же делать шаг возвращения из вершины x_k. Алгоритм заканчивает работу, когда на шаге возвращения достигается вершина x_1.

Следующие замечания могут помочь ускорить вышеприведенную процедуру прямого неявного перебора.

(а) При любом упорядочении вершин допустимые цвета j для вершины x_i удовлетворяют условию $j \leqslant i$ (если $i<q$). Это очевидно, так как вершине x_i предшествуют (при данном упорядочении) только $i-1$ вершин и, следовательно, никакой цвет $j>i$ не использовался. Таким образом, для вершины x_1 допустимым является только цвет 1, для x_2 — цвет 1 и цвет 2 (если x_2 смежна с x_1, то для x_2 допустим только цвет 2) и т. д.

[1]) В соответствии с заданным упорядочением.— *Прим. ред.*

(б) Из замечания (а) следует, что с вычислительной точки зрения выгодно располагать переменные в таком порядке, чтобы, например, первые ρ вершин образовывали наибольшую клику графа G. Это приведет к тому, что каждая вершина x_i $(1 \leqslant i \leqslant \rho)$ будет иметь только один допустимый цвет, т. е. цвет i, и алгоритм может закончить работу раньше, когда на шаге возвращения будет достигнута вершина x_ρ.

Хотя процедура неявного перебора, описанная в этом разделе, является примитивным древовидным поиском (в котором не нужно вычислять никакие оценки для ограничения ветвлений), тем не менее она нисколько не хуже других известных методов раскраски графов. Так, например, для графа с 30 вершинами, содержащего приблизительно третью часть множества всех ребер полного 30-вершинного графа, нахождение оптимальной раскраски с помощью алгоритма неявного перебора требует около 30 с машинного времени на вычислительной машине IBM 1130.

4. Приближенные алгоритмы раскрашивания

Существует много эвристических процедур раскрашивания графов, позволяющих находить хорошие приближения для хроматического числа графа в тех случаях, когда размеры графа слишком велики и получение оптимальной раскраски точными методами, упоминавшимися ранее, затруднительно. В настоящем разделе дается краткое описание одной из таких процедур и ряда ее разновидностей.

4.1. Последовательные методы, основанные на упорядочивании множества вершин

В этом простейшем из методов вершины вначале располагаются в порядке невозрастания их степеней.

Первая вершина окрашивается в цвет 1; затем список вершин просматривается сверху вниз (по невозрастанию степеней) и в цвет 1 окрашивается всякая вершина, которая не смежна с другой, уже окрашенной в этот цвет. Потом возвращаемся к первой в списке неокрашенной вершине, окрашиваем ее в цвет 2 и снова просматриваем список вершин сверху вниз, окрашивая в цвет 2 любую неокрашенную вершину, которая не соединена ребром с другой, уже окрашенной в цвет 2 вершиной. Аналогично действуем с цветами 3, 4 и т. д., пока не будут окрашены все вершины. Число использованных цветов будет тогда приближенным значением хроматического числа графа [23, 25].

Простая модификация описанной выше эвристической процедуры состоит в переупорядочивании неокрашенных вершин

после окраски каждой очередной вершины: оставшиеся неокрашенные вершины записываются в порядке невозрастания их «относительных» степеней, т. е. степеней в таком графе, который получается из данного после удаления окрашенных вершин (вместе с ребрами, инцидентными удаленным вершинам).

В этой процедуре молчаливо предполагалось, что если две вершины имеют одинаковые степени, то их взаимное положение в списке случайно. В таких ситуациях уточнение в размещении вершин можно осуществлять с помощью двухшаговых степеней $d_i^{(2)}$ вершин x_i, имеющих одинаковые степени (одинаковые 1-шаговые степени), где $d_i^{(2)}$ определяется как число маршрутов [1]) длины 2, исходящих из x_i. Эти вершины могут быть размещены тогда в соответствии с величинами степеней $d_i^{(2)}$. Если все-таки найдутся вершины, у которых совпадают и степени d_i, и степени $d_i^{(2)}$, то можно вычислить трехшаговые степени $d_i^{(3)}$ (определяемые аналогичным образом) и разместить вершины с учетом степеней $d_i^{(3)}$ и т. д.

Можно действовать иначе: размещать вершины сразу в соответствии с их степенями $d_i^{(2)}$ или степенями $d_i^{(3)}$ и применять тот же самый последовательный метод раскраски [24]. Таким образом, описанный выше метод раскрашивания очерчивает целый класс последовательных методов, каждый из которых связан с определенным способом упорядочивания вершин, либо статическим, т. е. фиксированным сразу для всей процедуры, либо динамическим, т. е. изменяющимся в процессе раскраски. Способ упорядочивания может базироваться на многих возможных критериях, зависящих от степеней вершин или от каких-либо других родственных характеристик. Результаты вычислений и сравнение последовательных методов раскрашивания для графов, выбранных случайным образом, приведены в работах Матулы, Марбле и Исааксона [16] и Вильямса [24]. Границы применимости этих эвристических методов демонстрируются у Митчема [17], показавшего, что можно построить графы, для которых любой из эвристических методов дает произвольно плохие оценки хроматического числа.

---

[1]) Если $\mathbf{A}$ — матрица смежности графа G с диагональными элементами, равными 0, то $d_i = \sum_{j=1}^{n} a_{ij}$. Двухшаговая степень определяется тогда по формуле $d_i^{(2)} = \sum_{j=1}^{n} a_{ij} d_j$, и вообще k-шаговая степень определяется по рекуррентной формуле:

$$d_i^{(k)} = \sum_{j=1}^{n} a_{ij} d_j^{(k-1)}.$$

5. Обобщения и приложения

Задача раскраски в том «чистом» виде, в каком она рассматривалась выше в настоящей главе, редко встречается на практике. Однако ее обобщения и разновидности (незначительно отличающиеся от нее) находят широкое применение в большом числе различных прикладных задач. Целью данного раздела является ознакомление читателя с несколькими наиболее часто встречающимися обобщениями. Список приложений, естественно, этими примерами не ограничивается.

5.1. Простая задача размещения (загрузки) [6]

Рассмотрим задачу размещения (загрузки) n каких-то предметов по ящикам. Пусть каждый предмет соответствует определенной вершине графа G. Всякий раз, когда два предмета x_i и x_j не могут быть размещены в одном ящике (например, когда предмет x_i может загрязнить предмет x_j), в граф G вводится ребро (x_i, x_j). Если ящики имеют неограниченную вместимость, так что в каждый из них можно поместить сколько угодно предметов, то задача нахождения наименьшего числа ящиков для размещения предметов эквивалентна задаче нахождения хроматического числа графа G; причем каждому ящику соответствует определенный «цвет», а предметы, окрашенные в один цвет, укладываются в один и тот же ящик.

(i) Ящики одинаковой вместимости. В действительности ящики обладают ограниченной вместимостью. Предположим, что вместимость одинакова у всех ящиков и равна Q. Это можно интерпретировать так: в один цвет окрашиваются не более чем Q вершин. В терминах 0-1-программирования (см. разд. 3 настоящей главы) данная задача загрузки может быть сформулирована следующим образом:

минимизировать выражение (4.14)
при ограничениях (4.11), (4.12) и

$$\sum_{i=1}^{n} \xi_{ij} \leqslant Q \text{ (для всех } j = 1, 2, \ldots, q). \tag{4.16}$$

Решением этой задачи является матрица $[\xi_{ij}^*]$, описывающая оптимальное *размещение* (распределение) *предметов по ящикам* (иначе говоря, *группировка предметов по цвету*). Число q есть верхняя оценка для числа ящиков.

(ii) Ящики имеют, вообще говоря, различные вместимости, и ящикам приписаны стоимости. Пусть j-й ящик имеет вместимость Q_j и стоимость v_j. Предположим еще, что предметам (верши-

нам графа) сопоставлены веса, скажем, i-му предмету соответствует вес w_i. Требуется так разместить предметы по ящикам, чтобы

(а) выполнялись условия, «накладываемые» графом G,

(б) удовлетворялись ограничения на вместимость ящиков,

(в) была наименьшей общая стоимость используемых ящиков.

Тогда задача примет следующий вид:

$$\text{минимизировать функцию } z = \sum_{j=1}^{q} \psi_j v_j \tag{4.17}$$

при ограничениях (4.11), (4.12) и

$$\left.\begin{aligned} \sum_{i=1}^{n} \xi_{ij} w_i &\leqslant Q_j, \qquad (4.18)\\ \sum_{i=1}^{n} \xi_{ij} &\leqslant L\psi_j \qquad (4.19) \end{aligned}\right\} \text{для всех } j = 1, \ldots, q,$$

где переменная ψ_j принимает значение 1, если в ящик j помещен какой-либо предмет, и равна 0 в противном случае. L — произвольное положительное целое число, большее, чем n, а q — общее число имеющихся ящиков. Ограничение (4.18) означает, что ни один из ящиков не перегружается, а ограничение (4.19) гарантирует выполнимость условия: если $\psi_{j_0} = 0$ (т. е. ящик j_0 пуст), то все ξ_{ij_0} ($i = 1, \ldots, n$) также равны 0.

В этом самом общем случае необходимо было ввести дополнительные переменные ψ_j, поскольку ящики (цвета́) нельзя штрафовать тем способом, который характеризуется выражением (4.14). Следует отметить, что чем более общей является задача, тем менее важным становится ее «раскрасочный» аспект. Так, например, в только что рассмотренной задаче можно выделить две подзадачи: (а) о «раскраске», соответствующую ограничению (4.12), и (б) о «ранце», определяемую ограничением [1]) (4.18). Два других ограничения, (4.11) и (4.19), являются ограничениями структурного характера.

Из-за этих двух взаимосвязанных аспектов общая задача раскраски значительно труднее для решения, чем «чистая» задача раскраски.

Две следующие прикладные задачи можно рассматривать как задачи о раскраске графа (или соответствующие обобщения задачи о раскраске).

---

[1]) *Задача о ранце* рассматривалась, например, в [10].

5.2. Составление графиков осмотра (проверки) [25] [2 4]

В задачах теории расписаний [1]) осмотры представляются в виде временны́х интервалов. Каждому осмотру можно сопоставить вершину некоторого графа, причем две любые вершины графа будут соединены ребром лишь тогда, когда соответствующие им осмотры нельзя осуществлять одновременно. Требуется составить такой график осмотра, который связан с наименьшими временны́ми затратами (с учетом приведенных выше ограничений на «совместимость» осмотров). Эта задача эквивалентна задаче о раскраске вершин графа с использованием наименьшего числа цветов. Хроматическое число графа как раз и соответствует осмотру, требующему наименьших временны́х затрат.

Если число осмотров, которые можно осуществлять в одно и то же время, ограничено (например, из-за размеров помещения), то приходим к задаче типа (i) из разд. 5.1, и Q в этом случае является числом помещений, где происходит осмотр.

5.3. Распределение ресурсов

Пусть для выполнения каких-то n работ надо распределить m имеющихся в наличии ресурсов. Считаем, что каждая из работ выполняется за некоторый (одинаковый для всех работ) промежуток времени и что для выполнения i-й работы требуется подмножество ресурсов S_i. Построим граф G: каждой работе соответствует определенная вершина графа, а ребро (x_i, x_j) существует в графе тогда и только тогда, когда для выполнения i-й и j-й работ требуется хотя бы один общий ресурс, т. е. когда $S_i \cap S_j \neq \varnothing$. Это означает, что i-я и j-я работы не могут выполняться одновременно. Раскраска графа G определяет тогда некоторое распределение ресурсов (по выполняемым работам), причем такое, что работы, соответствующие вершинам одного цвета, выполняются одновременно. Наилучшее использование ресурсов (т. е. выполнение всех n работ за наименьшее время) достигается при оптимальной раскраске вершин графа G.

6. Задачи

1. Для графа, показанного на рис. 4.5, вычислить верхние и нижние оценки хроматического числа, используя результаты, приведенные в разд. 2.1 и 2.2.

---

[1]) В отечественной литературе применяется также термин «задача календарного планирования».— *Прим. ред.*

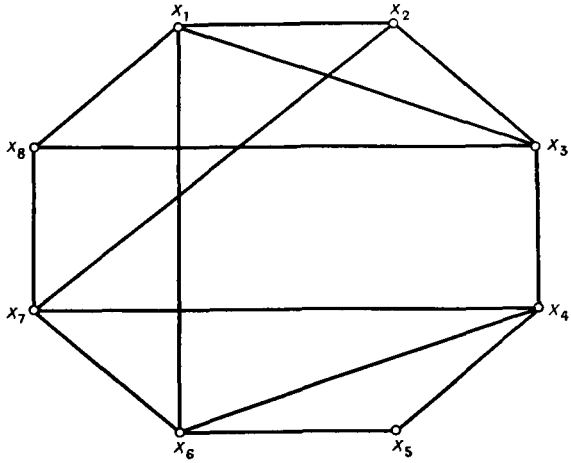


Рис. 4.5.

2. Используя алгоритмы из разд. 3.1 и 3.4, найти для графа на рис. 4.5 хроматическое число и соответствующую этому числу раскраску. Сравнить вычислительную сложность этих алгоритмов (для данного графа).

3. Для графа, изображенного на рис. 3.6 из гл. 3, найти хроматическое число и соответствующую раскраску, формулируя задачу о раскраске как ЗНП.

4. Применить последовательные методы, основанные на рассмотрении степеней d_i, $d_i^{(2)}$ и $d_i^{(3)}$, для получения «оптимальной» раскраски графа, приведенного на рис. 4.5.

5. Доказать, что хроматическое число каждого n-вершинного дерева ($n \geqslant 2$) равно 2.

6. Показать, что граф 2-хроматический тогда и только тогда, когда все его циклы имеют четные длины.

7. В плоскости проведено конечное число прямых линий; они разбили ее на конечное число областей. Показать, что достаточно 2 цвета для такой раскраски всех этих областей, когда любые две смежные области окрашиваются в разные цвета.

8. Граф на рис. 4.6 представляет схему электрических соединений; вершины соответствуют клеммам, ребра — прямым металлическим полоскам проводников. Для физически осуществимых соединений проводники не должны пересекать друг друга, поэтому необходимо распределить ребра по нескольким параллельным

платам, в каждой из которых проводники не пересекались бы. (Клеммы «доступны» на всех платах.)

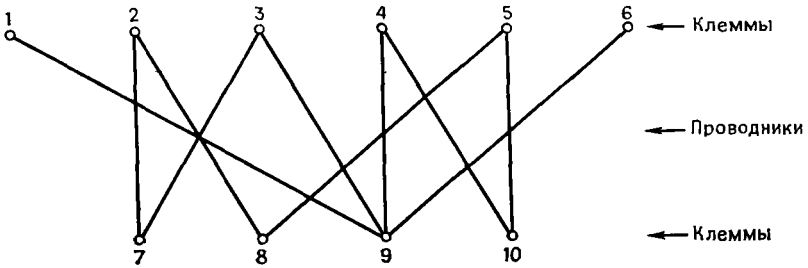


Рис. 4.6.

Определить наименьшее число плат, необходимых для реализации этих соединений (см. [14] и [7]).

9. На трех предприятиях выпускается одиннадцать видов изделий A, B, . . ., K, причем каждый вид производится только на одном из трех предприятий. Процент общих деталей и материалов, используемых в производстве каждых двух видов изделий, приводится в следующей матрице:

	A	B	C	D	E	F	G	H	I	J	K
A											
B	55	—									
C	60	80	—								
D	85	65	40	—							
E	45	90	35	80	—						
F	80	75	30	90	90	—					
G	50	30	70	35	65	30	—				
H	50	60	90	40	35	70	30	—			
I	60	45	40	80	30	70	20	25	—		
J	45	40	45	35	80	20	90	25	85	—	
K	80	70	85	30	35	40	25	75	75	60	—

Желательно, чтобы на одном предприятии выпускались изделия с большим процентом общих деталей и материалов, потребляемых при их производстве. Требуется так распределить производство изделий на предприятиях, чтобы минимум общих поставок для производства двух любых видов изделий на одном и том же предприятии был как можно больше.

7. Список литературы

1. Берж К., (1962), Теория графов и ее применения, ИЛ, М.
2. Bondy J. A. (1969), Bounds for the chromatic number of a graph, *J. of Combinatorial Theory*, 7, p. 96.
3. Brooks R. L. (1941), On colouring the nodes of a network, *Proc. Cambridge Philosophical Soc.*, 37, p. 194.
4. Brown J. R. (1972), Chromatic scheduling and the chromatic number problem, *Man. Sci.*, 19, p. 456.
5. Christofides N. (1971), An algorithm for the chromatic number of a graph, *The Computer J.*, 14, p. 38.
6. Eilon S., Christofides N. (1971), The loading problem, *Man. Sci.* 17, p. 259.
7. Even S. (1973), Algorithmic combinatorics, Macmillan, New York.
8. Geller D. P. (1970), Problem 5713, *American Mathematical Monthly*, 77, p. 85.
9. Gillian R. J. (1970 , The chromatic number of a graph, M. Sc. Thesis, Imperial College, London University.
10. Gilmore P. C., Gomory R. E. (1967), The theory and computation of knapsack functions, *Ops. Res.*, 15, p. 1045.
11. Graver J. E., Yackel J. K. (1968), Some graph theoretic results associated with Ramsey's theorem, *J. of Combinatorial Theory*, 4, p. 125.
12. Harary F. (1969), Graph Theory, Addison-Wessley, Reading, Massachusetts [Русский перевод: Харари Ф. (1973), Теория графов, М. «Мир».]
13. Heawood P. J. (1890), Map-colour theorems, *Quart. J. of Mathematics, Oxford Series*, 24, p. 322.
14. Lin C. L. (1968), Introduction to combinatorial mathematics, McGraw-Hill, New York.
15. Matula D. W. (1968), A min-max theorem for graphs with application to colouring, *J. of SIAM (Review)*, 10, p. 481.
16. Matula D. W., Marble G., Isaacson J. D. (1972), Graph colouring algorithms, Graph Theory and Computing, Read, Ed., Academic Press, New York. p. 109.
17. Mitchem J. (to appear), On various algorithms for estimating to the chromatic number.
18. Myers B. R., Lin R. (1972), A lower bound on the chromatic number of a graph, *Networks*, 1, p. 273.
19. Nordhous E. A., Gaddum J. W. (1956), On complementary graphs, *American Mathematical Monthly*, 63, p. 175.
20. Ore O. (1967), The four colour problem, Academic Press, New York.
21. Szekeres G., Wilf H. S. (1968), An inequality for the chromatic number of a graph, *J. of Combinatorial Theory*, 4, p. 1.
22. Tutte W. T. (B. Descartes) (1954), Solution of advanced problem N° 4526, *American Mathematical Monthly*, 61, p. 352.
23. Welsh D. J. A., Powell M. B. (1967), An upper bound on the chromatic number of a graph and its application to timetabling problems, *The Computer J.*, 10, p. 85.
24. Williams M. R. (1968), A graph theory model for the solution of timetables, Ph. D. Thesis, Iniversuty of Glasgow.
25. Wood D. C. (1969), A technique for colouring a graph applicable to large scale timetabling problems, *The Computer J.*, 12, p. 317.
26. Зыков А. А. (1949), О некоторых свойствах линейных комплексов, Матем сб., 24(66), 163 — 188.

Глава 5

РАЗМЕЩЕНИЕ ЦЕНТРОВ

1. Введение

В практической деятельности постоянно возникают задачи «наилучшего» размещения оборудования (или средств обслуживания) в сетях или графах. В частности, если граф представляет сеть дорог и вершины соответствуют отдельным районам, то можно поставить задачу оптимального размещения больниц, полицейских участков, пожарных частей и многих других крайне необходимых предприятий и служб. В таких случаях критерий оптимальности может состоять в минимизации расстояния (или времени проезда) от пункта обслуживания до самой отдаленной вершины графа, т. е. в оптимизации «наихудшего варианта». В более общей задаче требуется разместить несколько таких пунктов обслуживания (а не только один). При этом самая отдаленная вершина графа должна находиться по крайней мере от одного пункта обслуживания на минимально возможном расстоянии. К таким задачам относятся задачи размещения аварийных служб, и поэтому объективным требованием здесь является минимизация наибольшего расстояния от произвольной вершины графа до ближайшего к ней пункта обслуживания. По очевидным причинам задачи такого типа называются *минимаксными задачами размещения*. Полученные при решении этих задач места размещения пунктов обслуживания называются *центрами* графа.

В некоторых задачах размещения лучше всего было бы минимизировать сумму всех расстояний от вершин графа до центра обслуживания (если предполагать, что ищется место для размещения только одного такого пункта обслуживания). Такой критерий является наиболее подходящим, например, в задаче о размещении склада в сети дорог, где вершины представляют потребителей, обслуживаемых этим складом, или в задаче размещения телефонных станций в телефонной сети, где вершины представляют абонентов. Задачи такого типа вообще относятся к *минисуммным задачам размещения*, хотя целевая функция является часто не просто суммой расстояний, а суммой различных функций от расстояний (см. гл. 6). Места размещения пунктов обслуживания, полученные в результате решения минисуммной задачи, называются *медианами* графа.

Целью этой главы является рассмотрение минимаксной задачи размещения для взвешенных графов с весами c_{ij}, соответствующими дугам (и представляющими длины) и другими весами v_j, связанными с вершинами (и представляющими, скажем, размеры или важность объектов). Приводятся алгоритмы определения оптимального размещения центров в таких графах и результаты вычислений для графов средних размеров.

Минисуммная задача размещения рассматривается отдельно в следующей главе. Хотя эти две задачи, очевидно, связаны между собой, но, поскольку методы их решения различны, гл. 5 и 6 можно читать независимо друг от друга.

2. Разделения

Для любой вершины x_i графа $G = (X, \Gamma)$ пусть $R^o_\lambda(x_i)$ есть множество тех вершин x_j графа G, которые достижимы из вершины x_i с помощью путей со взвешенными длинами $v_j d(x_i, x_j)$, не превосходящими величины λ. Через $R^t_\lambda(x_i)$ будет обозначаться мно-

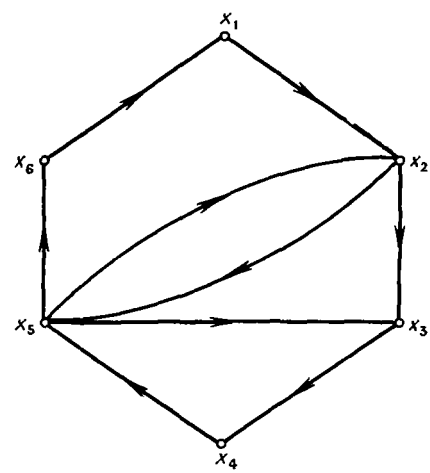


Рис. 5.1.

жество тех вершин x_j графа G, из которых вершина x_i может быть достигнута с использованием путей, имеющих взвешенные длины $v_j d(x_j, x_i) \leqslant \lambda$.

Таким образом,

$$R^o_\lambda(x_i) = \{x_j \mid v_j d(x_i, x_j) \leqslant \lambda,\ x_j \in X\}$$
и (5.1)
$$R^t_\lambda(x_i) = \{x_j \mid v_j d(x_j, x_i) \leqslant \lambda,\ x_j \in X\}.$$

Для каждой вершины x_i определим следующие два числа:

$$s_o(x_i) = \max_{x_j \in X} [v_j d(x_i, x_j)]$$

и

$$s_t(x_i) = \max_{x_j \in X} [v_j d(x_j, x_i)]. \qquad (5.2)$$

Числа $s_o(x_i)$ и $s_t(x_i)$ называются соответственно *числом внешнего разделения* и *числом внутреннего разделения вершины* x_i. Следует отметить, что $s_o(x_i)$ является наибольшим числом в строке x_i матрицы $D'(G)$, полученной в результате умножения каждого столбца j матрицы расстояний $D(G) = [d(x_i, x_j)]$ на v_j, и $s_t(x_i)$ является наибольшим числом в столбце x_i матрицы $D''(G)$, полученной после умножения каждой строки j матрицы расстояний $D(G)$ на v_j.

Рассмотрим в качестве примера ориентированный граф, изображенный на рис. 5.1, и предположим, что все веса вершин и дуг графа равны единице. Матрица расстояний графа имеет вид

$D(G) =$

	x_1	x_2	x_3	x_4	x_5	x_6	$s_o(x_i)$
x_1	0	1	2	3	2	3	3
x_2	3	0	1	2	1	2	3
x_3	4	3	0	1	2	3	4
x_4	3	2	2	0	1	2	3
x_5	2	1	1	2	0	1	2*
x_6	1	2	3	4	3	0	4
$s_t(x_i)$	4	3*	3*	4	3*	3*	

Числа внешних и внутренних разделений приведены в присоединенных к матрице столбце и строке соответственно.

Если λ_o — наименьшая длина λ, такая, что для вершины x_i

$$R^o_\lambda(x_i) = X$$

(т. е. все вершины графа G достижимы из x_i с использованием путей, взвешенные длины которых не превосходят λ_o, причем λ_o — наименьшее из таких чисел), то из соотношений (5.1) и (5.2) следует равенство

$$s_o(x_i) = \lambda_o. \qquad (5.3)$$

Аналогично, если λ_t — такая наименьшая длина λ, что

$$R_\lambda^t(x_i) = X,$$

то $s_t(x_i) = \lambda_t$.

Совершенно очевидно, что у графа G числа внешнего и внутреннего разделений любой вершины конечны только тогда, когда граф сильно связный, т. е. когда каждая вершина достижима из всякой другой вершины.

3. Центр и радиус

Вершина x_o^*, для которой

$$s_o(x_o^*) = \min_{x_i \in X} [s_o(x_i)], \qquad (5.4\,(а))$$

называется *внешним центром* графа G; и аналогично вершина x_t^*, для которой

$$s_t(x_t^*) = \min_{x_i \in X} [s_t(x_i)], \qquad (5.4\,(б))$$

называется *внутренним центром* графа G.

У графа может быть несколько (больше, чем один) внешних и внутренних центров. Таким образом они образуют множества внешних и внутренних центров соответственно.

Число внешнего разделения вершины x_o^*, являющейся внешним центром, называется *внешним радиусом*: $\rho_0 = s_o(x_o^*)$; число внутреннего разделения внутреннего центра называется *внутренним радиусом*: $\rho_t = s_t(x_t^*)$.

У графа, изображенного на рис. 5.1, с матрицей расстояний, приведенной выше, имеются только один внешний центр (вершина x_5) и четыре внутренних центра, образующих множество $\{x_2, x_3, x_5, x_6\}$. Внешний радиус графа равен 2, а внутренний 3.

3.1. Размещение аварийных служб и пунктов обслуживания

Рассмотрим задачу обслуживания нескольких жилых районов или населенных пунктов (связанных между собой дорожной сетью) каким-либо одним пунктом обслуживания (например, одной больницей, или полицейским участком, или пожарным депо). По некоторым причинам (например, наличие людских ресурсов или другие удобства) пункт обслуживания должен быть размещен в одном из этих районов, а не в произвольной точке какой-либо дороги.

Предположим, что «длины» c_{ij} дуг графа G (вершины которого соответствуют районам, а дуги — дорогам) образуют матрицу времен проезда между этими районами. Эта матрица необязательно симметрическая, т. е., вообще говоря, $c_{ij} \neq c_{ji}$, поскольку

время проезда может зависеть от уклонов на дороге, наличия улиц с односторонним движением и т. д.

В случае размещения полицейского участка или пожарного депо интересуются временем, необходимым для проезда в наиболее отдаленный из этих районов. Следовательно, задача размещения полицейского участка (или пожарного депо) состоит в минимизации этого времени. Задача становится более реалистичной, если каждой вершине графа приписывается вес (v_j), представляющий вероятность потребности данного района в соответствующем обслуживании. Эти веса, например, могут быть пропорциональны численности населения каждого района. Вершина, которая минимизирует время проезда до самого отдаленного района, является внешним центром графа.

В случае размещения больницы интересуются временем, необходимым для проезда машины скорой помощи в самый отдаленный район и возвращения ее в больницу. Если определить число внешне-внутреннего разделения вершины x_i с помощью равенства

$$s_{ot}(x_i) = \max_{x_j \in X} \{v_j [d(x_i, x_j) + d(x_j, x_i)]\},$$

то вершину x_{ot}^*, на которой достигается минимум выражения

$$s_{ot}(x_i),$$

можно назвать внешне-внутренним центром.

Для графа, изображенного на рис. 5.1, с матрицей расстояний $D(G)$, приведенной ранее, внешне-внутренним центром является вершина x_5. Внешне-внутренний радиус равен 5.

4. Абсолютный центр

Соотношение (5.3) определяют числа разделения для любой вершины $x_i \in X$. Мы обобщим теперь это определение на случай

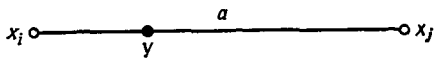


Рис. 5.2. Размещение на дуге.

«искусственных» точек, которые можно помещать на дугах. Итак, если $a = (x_i, x_j)$ (см. рис. 5.2) представляет дугу графа с весом (длиной) c_{ij}, то точка y, помещаемая на этой дуге, может быть определена посредством задания длины $l(x_i, y)$ участка (x_i, y) причем должно выполняться равенство

$$l(x_i, y) + l(y, x_j) = c_{ij}. \tag{5.5}$$

Числа разделения $s_o(y)$ и $s_t(y)$ точки y независимо от того, является она вершиной графа G или искусственной точкой дуги

графа G, определяются следующим образом:

$$s_o(y) = \max_{x_i \in X} [v_i d(y, x_i)], \tag{5.6}$$

а для $s_t(y)$ соответствующее выражение получается из соотношения (5.2).

Точка y_o^*, для которой

$$s_o(y_o^*) = \min_{y \text{ из } G} [s_o(y)], \tag{5.7}$$

называется *абсолютным внешним центром* графа; и аналогично определяется y_t^* — *абсолютный внутренний центр*.

Число внешнего разделения абсолютного внешнего центра называется *абсолютным внешним радиусом*: $r_o = s_o(y_o^*)$, и число внутреннего разделения абсолютного внутреннего центра называется *абсолютным внутренним радиусом*: $r_t = s_t(y_t^*)$.

Пример. Рассмотрим неориентированный граф G, показанный на рис. 5.3, где все веса вершин и длины ребер равны единице.

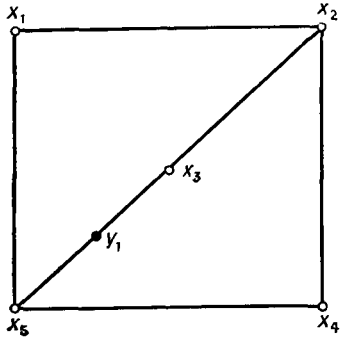


Рис. 5.3. Абсолютный центр.

Поскольку граф неориентированный, то числа внешних и внутренних разделений одинаковые (для одной и той же вершины).

Матрица расстояний графа имеет вид

$D(G)$

	x_1	x_2	x_3	x_4	x_5
x_1	0	1	2	2	1
x_2	1	0	1	1	2
x_3	2	1	0	2	1
x_4	2	1	2	0	1
x_5	1	2	1	1	0

Нетрудно видеть, что центром является каждая вершина и что радиус равен 2.

Если теперь выбрать точку y_1 на ребре (x_3, x_5) так, что $l(x_5, y_1) = \frac{1}{2}$ $\left(\text{и, следовательно, } l(y_1, x_3) = \frac{1}{2}\right)$, то расстояния от этой точки до вершин графа даются таблицей

	x_1	x_2	x_3	x_4	x_5
y_1	$1\frac{1}{2}$	$1\frac{1}{2}$	$\frac{1}{2}$	$1\frac{1}{2}$	$\frac{1}{2}$

Значит, наибольшее число разделения равно $1\frac{1}{2}$. Таким образом, точка y_1 более «центральна», чем любая из вершин графа G. В действительности точка y_1 является абсолютным центром графа G — так же, как и все точки, расположенные в середине ребер (x_1, x_2), (x_2, x_3), (x_5, x_4), (x_4, x_2) и (x_5, x_1). Ни одна из вершин графа не является абсолютным центром. Таким образом, в общем случае может быть один или больше абсолютных центров, которые располагаются или в вершинах, или на дугах графа.

5. Алгоритмы нахождения абсолютных центров

Центры и радиусы графа можно найти непосредственно из матрицы взвешенных расстояний, как было показано в разд. 2 и 3. Приведем два метода нахождения абсолютного центра графа и проиллюстрируем эти методы на примере.

5.1. Метод Хакими [7]

Этот метод очень прост и для неориентированного графа состоит в следующем (для ориентированного графа метод остается таким же, надо только каждое «неориентированное» понятие заменить его «ориентированным двойником»).

(i) Для каждого ребра a_k графа найти точки (или точку) y_k^* на a_k, которые имеют наименьшее число разделения.

(ii) Из всех точек y_k^* $(k = 1, 2, \ldots, m)$ в качестве абсолютного центра графа G выбрать точку с наименьшим числом разделения.

Первый шаг метода осуществляется следующим образом.

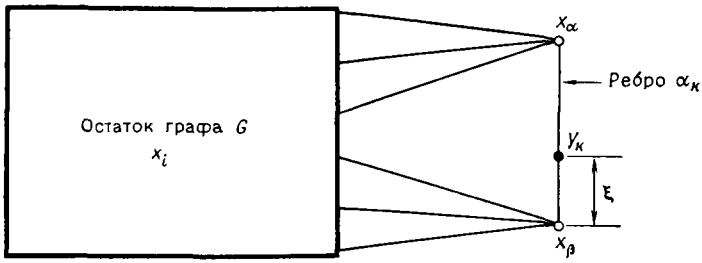


Рис. 5.4.

Возьмем ребро a_k графа G (см. рис. 5.4). Имеем (из соотношения (5.6)):

$$s(y_k) = \max_{x_i \in X} [v_i d(y_k, x_i)] =$$
$$= \max_{x_i \in X} [v_i \min \{l(y_k, x_\beta) + d(x_\beta, x_i), l(y_k, x_\alpha) + d(x_\alpha, x_i)\}], \quad (5.8)$$

поскольку расстояние $d(y_k, x_i)$ равно либо длине маршрута, проходящего через вершину x_α, либо длине маршрута, проходящего через x_β, $l(y_k, x_\alpha)$ и $l(y_k, x_\beta)$ являются длинами соответствующих частей ребра a_k.

Пусть $l(y_k, x_\beta) = \xi$. Поскольку $l(y_k, x_\alpha) = c_{\alpha\beta} - l(y_k, x_\beta) = c_{\alpha\beta} - \xi$, то соотношение (5.8) примет вид

$$s(y_k) = \max_{x_i \in X} \min [v_i \{\xi + d(x_\beta, x_i)\}, v_i \{c_{\alpha\beta} + d(x_\alpha, x_i) - \xi\}]. \quad (5.9)$$

Для фиксированной вершины x_i и при каждом значении ξ $(0 \leqslant \xi \leqslant c_{\alpha\beta})$ можно найти наименьшие значения выражений, заключенных в соотношении (5.9) в квадратные скобки. Для этого выпишем отдельно два указанных выражения:

$$T_i = v_i \{\xi + d(x_\beta, x_i)\},$$
$$T_i' = v_i \{c_{\alpha\beta} + d(x_\alpha, x_i) - \xi\} \quad (5.10)$$

и, рассматривая их относительно ξ, строим нижнюю «огибающую» для соответствующих им прямых линий [1]).

Повторяя эту процедуру для всех вершин $x_i \in X$, мы построим на одном и том же чертеже все остальные нижние «огибающие». Далее вычертим верхнюю «огибающую» для всех ранее полученных нижних «огибающих», которая (в силу соотношения (5.9)) дает числа разделения $s(y_k)$ для всех значений параметра ξ, т. е.

[1]) В действительности эта «огибающая» представляет собой ломаную линию, состоящую из двух «нижних» лучей (от точки пересечения) рассматриваемых прямых линий.— *Прим. ред.*

для всех точек y_k ребра a_k. Построенная огибающая (она составлена из линейных кусков) может иметь несколько минимумов. Точка y_k, соответствующая наименьшему из этих минимумов, является (в силу соотношения (5.7)) абсолютным центром y_k^*, отвечающим дополнительному ограничению: он должен лежать на ребре a_k. Абсолютным центром графа будет такая точка y_k^*, которой соответствует наименьший из минимумов, определяющих указанные выше абсолютные центры на ребрах a_k ($k = 1, 2, \ldots, m$).

5.2. Размещение аварийных служб (общий случай)

Рассмотрим еще раз задачу об обслуживании нескольких районов каким-либо одним пунктом обслуживания (одной больницей, или пожарным депо, или полицейским участком). Если опущено ограничение, состоящее в том, что пункт обслуживания должен размещаться в каком-то из жилых районов (см. разд. 3.1), то оптимальным размещением пункта обслуживания будет его размещение в любом абсолютном центре соответствующего графа.

Рассмотрим, например, неориентированный граф, приведенный на рис. 5.5. Пусть вершины графа соответствуют жилым районам, а длина ребра (x_i, x_j) равна времени (в минутах), необходимому для проезда из района x_i в район x_j. Предположим, что вершины графа имеют единичные «веса» и матрица «расстояний» (времен) такова:

$D(G) =$

	x_1	x_2	x_3	x_4	x_5
x_1	0	8	8	6	2
x_2	8	0	2	10	6
x_3	8	2	0	12	8
x_4	6	10	12	0	4
x_5	2	6	8	4	0

Очевидно, что центром графа является либо вершина x_1, либо x_5 и что радиус графа равен 8.

Сначала возьмем ребро a_1 (рис. 5.5), и пусть расстояние ξ измеряется от вершины x_3 до вершины x_1.

Выражения T_i и T_i' из соотношений (5.10) (для $i = 1, 2, \ldots, 5$) имеют в нашем случае такой вид:

$$T_1 = \xi + d(x_3, x_1) = \xi + 8,$$
$$T_1' = c_{3,1} + d(x_1, x_1) - \xi = 8 - \xi,$$

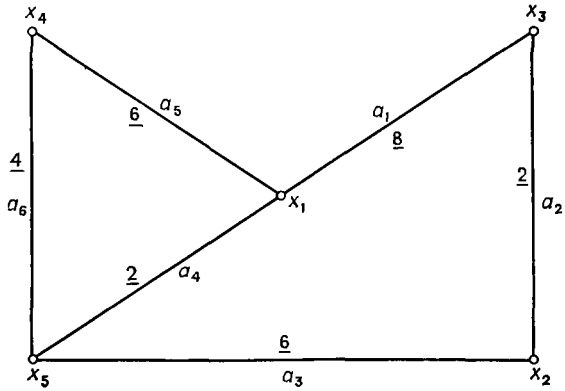


Рис. 5.5. Граф из примера разд. 5.2.

аналогично

$$T_2 = \xi + 2, \qquad T_2' = 16 - \xi,$$
$$T_3 = \xi, \qquad T_3' = 16 - \xi,$$
$$T_4 = \xi + 12, \qquad T_4' = 14 - \xi,$$
$$T_5 = \xi + 8, \qquad T_5' = 10 - \xi.$$

Графики этих функций изображены на рис. 5.6а, для $0 \leqslant \xi \leqslant 8$. Имеются два локальных абсолютных центра y_1^* на расстоянии 6 и 8 минут от x_3. Число разделения этих двух центров равно 8 минутам.

Аналогично строятся графики для ребер a_2, a_3, . . ., см. рис. 5.6 б—е. Абсолютный центр (y^*) этого графа есть такой локальный центр, который имеет наименьшее число разделения. Он, как легко видеть, находится на ребре a_3 (рис. 5.6в) в двух минутах от x_5. Абсолютный радиус r (соответствующий y^*), как видно из рис. 5.6, равен 6.

5.3. Модифицированный метод Хакими

В описанном выше методе Хакими поиск локального центра осуществляется вдоль всего ребра графа G. Если в графе много ребер, то время вычисления, требуемое для поиска, может оказаться чрезвычайно большим. Рассматриваемая в данном разделе модификация метода Хакими состоит в вычислении верхних и нижних оценок абсолютных локальных радиусов, соответствующих локальным центрам ребер, и в использовании полученных оценок для уменьшения числа ребер, участвующих в поиске.

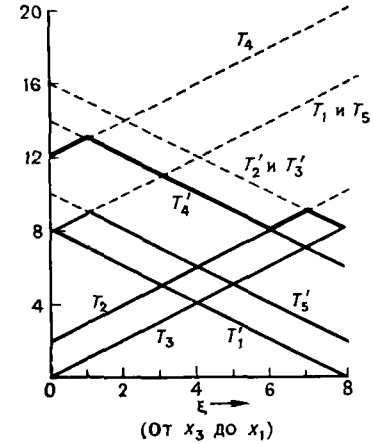


Рис. 5.6а. Размещение на ребре a_1.

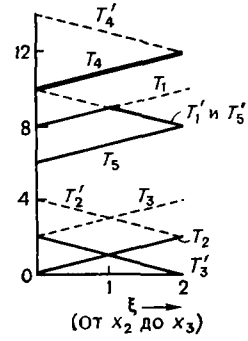


Рис. 5.6б. Размещение на ребре a_2.

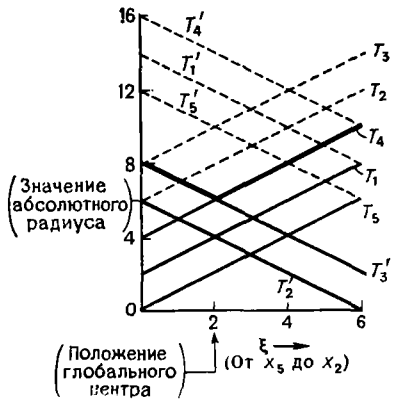


Рис. 5.6в. Размещение на ребре a_3.

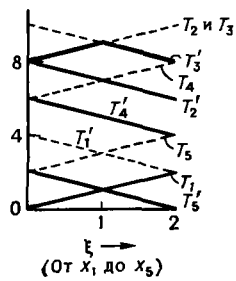


Рис. 5.6г. Размещение на ребре a_4.

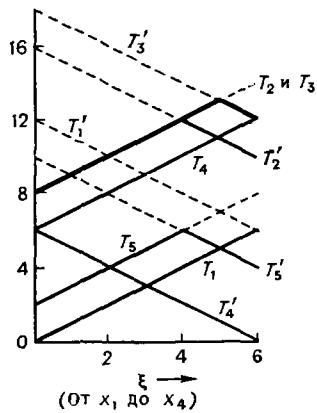


Рис. 5.6д. Размещение на ребре a_5.

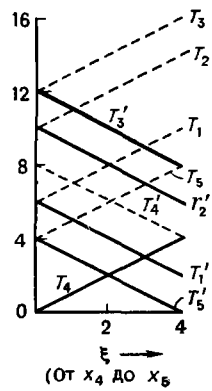


Рис. 5.6е. Размещение на ребре a_6.

Всякому локальному центру, расположенному на ребре (x_i, x_j), соответствует (как видно из соотношения (5.8), если положить все длины l равными нулю) его абсолютный локальный радиус (скажем, r_{ij}), который не меньше, чем p_{ij}, где

$$p_{ij} = \max_{x_s \in X} [v_s \min \{d(x_s, x_i), d(x_s, x_j)\}]. \qquad (5.11)$$

Таким образом, p_{ij} есть *нижняя* оценка абсолютного радиуса графа, если абсолютный центр лежит на ребре (x_i, x_j). Следовательно, величина

$$P = \max_{(x_i, x_j) \in A} [p_{ij}] \qquad (5.12)$$

(где A — множество ребер графа) является обоснованной нижней оценкой абсолютного радиуса графа.

Допустим, что абсолютный центр расположен в середине ребра (x_i, x_j). Тогда, в силу соотношения (5.8), абсолютный радиус равен $p_{ij} + \frac{1}{2} v_{s*} c_{ij}$, где v_{s*} — вес той вершины x_s, в которой достигается наибольшее значение величины p_{ij}, указанной в соотношении (5.11). Следовательно,

$$H = \min_{(x_i, x_j) \in A} [p_{ij} + \tfrac{1}{2} v_{s*} c_{ij}] \qquad (5.13)$$

является обоснованной *верхней* оценкой абсолютного радиуса графа. Таким образом, всякое ребро (x_{i_o}, x_{j_o}), для которого $p_{i_o j_o} \geqslant H$, может не рассматриваться при поиске абсолютного центра.

Для графа, приведенного на рис. 5.5, у которого $v_i = 1$ для всех $x_i \in X$, получаем (используя формулу (5.11)) следующие результаты:

центр на $a_1 = (x_3, x_1)$ дает $p_{3,1} = \max(2, 6, 2) = 6$,
центр на $a_2 = (x_2, x_3)$ дает $p_{2,3} = \max(8, 10, 6) = 10$,
центр на $a_3 = (x_5, x_2)$ дает $p_{5,2} = \max(2, 2, 4) = 4$,
центр на $a_4 = (x_1, x_5)$ дает $p_{1,5} = \max(6, 8, 4) = 8$,
центр на $a_5 = (x_1, x_4)$ дает $p_{1,4} = \max(8, 8, 2) = 8$,
центр на $a_6 = (x_4, x_5)$ дает $p_{4,5} = \max(2, 6, 8) = 8$.

Верхняя оценка H равна, следовательно (в силу соотношения (5.13)),

$$\min(6 + 4,\ 10 + 1,\ 4 + 3,\ 8 + 1,\ 8+3,\ 8+2) = 7,$$

и поэтому ребра a_2, a_4, a_5 и a_6, для которых $p_{ij} > 7$, могут быть исключены из списка кандидатов на размещение абсолютного центра. Поиск абсолютного центра нужно вести только на двух оставшихся ребрах (так, как это было продемонстрировано раньше

ше и показано на рис. 5.6 (а) и 5.6 (в)). Используя верхние и нижние оценки, удалось уменьшить поиск в 3 раза. Наименьшая нижняя оценка P, вычисленная по формуле (5.12), равна

$$P = \min(6,\ 10,\ 4,\ 8,\ 8,\ 8) = 4.$$

Эта оценка не является точной, поскольку истинная величина абсолютного радиуса r данного графа (найденная в разд. 5.2) равна 6.

5.4. Итерационный метод

Чтобы сделать изложение более простым, мы опишем данный метод для случая неориентированных графов. Замена каждого «неориентированного» понятия его «ориентированным двойником» не приводит к какому-либо коренному изменению метода.

Пусть $Q_\lambda(x_i)$ — множество всех таких точек y, лежащих на ребрах графа G, из которых вершина x_i достижима со взвешенным расстоянием, не превосходящим λ. Таким образом,

$$Q_\lambda(x_i) = \{y \mid v_i d(y, x_i) \leqslant \lambda,\ y \text{ — точка графа } G\}. \qquad (5.14)$$

Это выражение похоже на те (см. (5.1)), с помощью которых были определены множества $R_\lambda^0(x_i)$ и $R_\lambda^t(x_i)$; отличие состоит в том, что теперь y — любая точка графа G, а необязательно его вершина. Абсолютный радиус r, очевидно, является наименьшим значением λ, при котором из некоторой точки y графа G все вершины графа могут быть достигнуты со взвешенным расстоянием, меньшим или равным λ. Следовательно, r есть такое наименьшее значение λ (скажем, $\lambda_{\min}$), что

$$\bigcap_{x_i \in X} [Q_\lambda(x_i)] \equiv Q_\lambda(x_1) \cap Q_\lambda(x_2) \cap \ldots \cap Q_\lambda(x_n) \neq \varnothing. \qquad (5.15)$$

Поэтому можно начать с произвольного небольшого значения λ, строить множества $Q_\lambda(x_i)$ для всех $i = 1, 2, \ldots, n$ и проверять, выполняется ли соотношение (5.15). Если оно не выполняется, то надо увеличить немного величину λ, заново построить множества $Q_\lambda(x_i)$ (при новом значении λ) и опять проверить, не выполняется ли соотношение (5.15). Эту процедуру можно повторять до тех пор, пока не будет выполняться (5.15). Полученная таким образом величина λ принимается за абсолютный радиус r графа G. Более того, поскольку приращения величины λ малы, то пересечение

$$\bigcap_{x_i \in X} [Q_\lambda(x_i)] \qquad (5.16)$$

при завершении процесса итерации будет содержать только одну точку (этого для практических целей достаточно), и она является как раз абсолютным центром y^*. (Возможно, что будет не одна точка, если существует более чем один абсолютный центр.)

Поскольку $d(x_i, x_j)$ есть наикратчайшее расстояние между двумя произвольными вершинами x_i и x_j, то совершенно очевидно, что если λ меньше половины взвешенного расстояния между x_i и x_j, т. е.

$$\lambda \leqslant \frac{v_i v_j}{v_i + v_j} d(x_i, x_j), \quad \text{то} \quad Q_\lambda(x_i) \cap Q_\lambda(x_j) = \varnothing$$

и полное пересечение множеств в выражении (5.16) пусто.

Следовательно, итерационный процесс поиска абсолютного центра можно начать со значения λ, равного

$$\lambda_0 = \max_{x_j i^{-1} x \in X} \left[\frac{v_i v_j}{v_i + v_j} d(x_i, x_j) \right], \tag{5.17}$$

поскольку радиус r должен быть не меньше λ_0. С этой точки зрения, значение $\delta = 2\lambda_0$ можно назвать диаметром графа. Нужно, однако, отметить, что совсем необязательно диаметр графа в два раза больше значения абсолютного радиуса (определенного в разд. 4, см. в связи с этим задачу 5.11).

Детальное описание более общего алгоритма, частью которого является рассмотренный здесь алгоритм, будет дано позже, в разд. 8.

6. Кратные центры (p-центры)

Понятие центра графа допускает следующее обобщение: можно рассматривать не отдельную точку (центр), а множество из p точек, которые образуют кратный центр (p-центр).

Пусть X_p — подмножество (содержащее p вершин) множества X вершин графа $G = (X, \Gamma)$. Через $d(X_p, x_i)$ будем обозначать наикратчайшее из расстояний между вершинами множества X_p и вершиной x_i, т. е.

$$d(X_p, x_i) = \min_{x_j \in X_p} [d(x_j, x_i)].$$

1277

Аналогично, символом $d(x_i, X_p)$ обозначается $\min\limits_{x_j \in X_p} [d(x_i, x_j)]$.

Подобно тому, как определялись числа разделения вершин (см. разд. 3), мы можем определить числа разделения для множеств вершин:

$$s_o(X_p) = \max_{x_j \in X} [v_j d(X_p, x_j)]$$

и

$$s_i(X_p) = \max_{x_j \in X} [v_j d(x_j, X_p)], \tag{5.18}$$

где $s_o(X_p)$ и $s_i(X_p)$ — *числа внешнего и внутреннего разделения множества* X_p.

Множество X^*_{po}, для которого

$$s_o(X^*_{po}) = \min_{X_p \subseteq X} [s_o(X_p)], \tag{5.19}$$

называется *p-кратным внешним центром* графа G; аналогично определяется *p-кратный внутренний центр* X^*_{pi}.

В разд. 2 и 3 указывалось, что центры графа легко могут быть получены из матрицы взвешенных расстояний. Однако находить таким же способом (полным перебором) p-центр можно лишь для небольших графов и для небольших значений величины p. При таком подходе надо построить всевозможные множества вершин $X_p \subseteq X$, содержащие p вершин, а затем, используя формулы (5.18) и (5.19), непосредственно найти множества X^*_{po} и X^*_{pi}, образующие p-центры. Если предположить, что матрица расстояний уже известна, то непосредственное применение соотношений (5.18) и (5.19) потребует выполнить $p \cdot (n-p) \cdot \binom{n}{p}$ сравнений. Это число при $n = 100$ и $p = 5$ равно $3{,}58 \cdot 10^{10}$, что значительно превышает возможности существующих ЭВМ.

6.1. Задача размещения нескольких пунктов обслуживания

В разд. 3.1 была рассмотрена задача размещения одной больницы (или одного полицейского участка, или одного пожарного депо) в графе, представляющем реальную сеть дорог. Однако очень часто имеет место такая ситуация, когда одного пункта обслуживания недостаточно, поскольку он не в состоянии обслужить все поступающие вызовы, и тогда возникает задача о наилучшем размещении нескольких таких пунктов обслуживания. Эту задачу можно сформулировать так.

Найти наименьшее число пожарных депо (например) и такое их размещение, чтобы расстояние от каждого жилого района до ближайшего к нему пожарного депо не превышало наперед заданной величины. Если же число пожарных депо известно, то требуется разместить их так, чтобы было минимально возможным расстояние от любого района до ближайшего к нему депо.

Если предположить, что пожарные должны размещаться в вершинах соответствующего графа G, то задача будет состоять в нахождении p-центров графа для $p = 1, 2, 3, \ldots$ и т. д. до тех пор, пока число разделения p-центра не станет меньше или равно заданному расстоянию. Полученное (последнее) значение числа p будет наименьшим числом пожарных депо, а p-центр — их оптимальным размещением, удовлетворяющим предъявляемым требованиям.

Алгоритм нахождения p-центров является частным случаем алгоритма решения более общей задачи, состоящей в определении

p-центров, располагающихся, вообще говоря, не в вершинах графа. Рассмотрение общего алгоритма поиска p-центров мы отложим до разд. 8.5, пока не будет исследована соответствующая более общая задача.

7. Абсолютные p-центры

Если ограничение, согласно которому точки p-центра должны лежать в вершинах графа, снято и допускается размещение точек на дугах, то получающееся при этом (более общее) множество из p точек называется *абсолютным p-центром*. Таким образом, тот объект, который в разд. 5 был назван абсолютным центром, в соответствии с настоящей терминологией можно назвать абсолютным 1-центром.

Задача нахождения абсолютного p-центра может быть сформулирована следующим образом.

(а) Найти оптимальное размещение в любых точках графа заданного числа (например, p) центров при условии, что расстояние (время) до самой отдаленной вершины от ближайшего к ней центра является минимально возможным.

Вторая задача, очень близкая к задаче (а) и которая, как будет показано позже, может быть решена тем же методом, что и задача (а), состоит в следующем.

(б) Для заданного «критического» расстояния найти такое наименьшее число центров и такое их размещение, чтобы все вершины графа лежали в пределах этого критического расстояния (по крайней мере каждая вершина — от ближайшего к ней центра).

Это — общая задача определения абсолютных p-центров. Именно она наиболее часто встречается на практике. Однако решать ее гораздо труднее, чем какой-либо из ее «ограниченных» вариантов. Метод Хакими [7, 8], приведенный в разд. 5 и предназначенный для решения задачи с одним абсолютным центром, не может быть обобщен на случай абсолютных p-центров. Для нахождения таких центров Сингер [12] предложил некоторый эвристический метод.

В данном разделе мы познакомимся с итерационным алгоритмом решения задачи об абсолютных p-центрах графа. Из приведенных далее результатов вычислений видно, что этот алгоритм является быстро сходящимся. Метод обладает следующими двумя преимуществами:

(i) Процесс можно закончить сразу же, как только достигнута необходимая «точность» в расположении центров.

(ii) Метод легко видоизменить таким образом, чтобы можно было находить решения, близкие к оптимальному, и, следовательно, проводить анализ устойчивости решения.

Ради упрощения обозначений мы будем рассматривать только неориентированные графы. Распространение полученных результатов на ориентированные графы осуществляется очевидным образом.

Пусть Y_p — произвольное множество каких-либо p точек на графе G. Число разделения $s(Y_p)$ множества Y_p определяется так:

$$s(Y_p) = \min_{x_j \in X} [v_j d(Y_p, x_j)], \tag{5.20}$$

где

$$d(Y_p, x_j) = \min_{y_i \in Y_p} [d(y_i, x_j)].$$

Абсолютный p-центр графа G определяется как множество точек Y_p^*, для которого

$$s(Y_p^*) = \min_{Y_p \text{ на } G} [s(Y_p)]. \tag{5.21}$$

8. Алгоритм нахождения абсолютных p-центров

Рассмотрим по очереди каждую вершину x_i и «углубимся» по всем возможным маршрутам, выходящим из нее, на расстояние $\delta_i = \lambda/v_i$, где λ — заданная константа, которую мы будем называть *константой «проникновения»*.

Пусть $Q_\lambda(x_i)$ — множество всех точек y на графе G, из которых вершина x_i достижима в пределах расстояния δ_i при заданном значении λ. Множества $Q_\lambda(x_i)$ определяются с помощью соотношения (5.14) из разд. 5.2. Эти множества весьма легко можно построить, применяя алгоритм, подобный алгоритму Дейкстра [4] (позволяющему находить кратчайшие пути в графе, см. гл. 8).

Определим *область* Φ_λ как множество таких точек y на графе G, что из каждой точки y достижимо в пределах расстояния [1]) δ_i (при заданном λ) одно и то же множество вершин графа G. Область может быть, например, частью ребра или может содержать только одну точку. Рассмотрим в качестве примера граф, изображенный на рис. 5.7. Пусть все веса вершин равны единице, а длины ребер равны $c_{1,2} = 4$, $c_{2,3} = 8$ и $c_{3,1} = 6$. Возьмем $\lambda = 3$. Тогда $\delta_i = 3$ для всякого i и существует шесть различных областей.

---

[1]) В дальнейшем (в тех случаях, когда не могут возникнуть недоразумения) слова «в пределах расстояния... при заданном λ» опускаются.— *Прим. ред.*

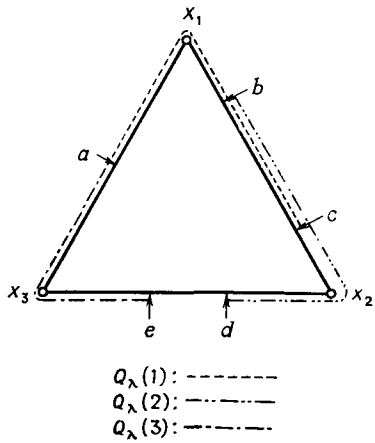


Рис. 5.7. Области, образованные при $\lambda = 3$.

Область 1. Точка a (единственная точка, из которой в пределах расстояния, равного 3, достижимы вершины x_1 и x_3).

Область 2. Отрезок ребра (x_1, x_2) от b до c. (Из любой точки этой области достижимы вершины x_1 и x_2.)

Область 3. Отрезки (a, x_1) и (x_1, b) соответствующих ребер (из точек этой области достижима только вершина x_1).

Область 4. Отрезки (a, x_3) и (x_3, e) соответствующих ребер (из каждой точки этой области достижима только x_3).

Область 5. Отрезки (c, x_2) и (x_2, d) (из точек этой области достижима только x_2).

Область 6. Отрезок ребра (x_2, x_3) от e до d (из точек этой области не достижима никакая вершина).

В общем случае все области Φ_λ можно следующим образом построить из достижимых множеств Q_λ.

Области, из которых не достижимы никакие вершины [1]), описываются соотношением

$$\Phi_\lambda(0) = \{y \mid y \text{ на } G\} - \bigcup_i Q_\lambda(x_i), \tag{5.22}$$

где второй член исключает все области графа G, из которых можно достигнуть [1]) хотя бы одну вершину x_i.

Области, из которых можно достигнуть [1]) ровно t вершин $x_{i_1}, x_{i_2}, \ldots, x_{i_t}$ (для любого $t = 1, \ldots, n$), определяются сле-

[1]) Всюду в этих рассмотрениях достижимость берется «в пределах заданного расстояния δ_i».— *Прим. ред.*

дующим выражением:

$$\Phi_\lambda(x_{i_1}, x_{i_2}, \ldots, x_{i_t}) = \bigcap_{q=1,\ldots,t} Q_\lambda(x_{i_q}) - \{[\bigcap_{q=1,\ldots,t} Q_\lambda(x_{i_q})] \cap [\bigcup_{q=t+1,\ldots,n} Q_\lambda(x_{i_q})]\}, \quad (5.23)$$

где второй член исключает такие области, из которых достижимы вершины $x_{i_1}, x_{i_2}, \ldots, x_{i_t}$ и еще хотя бы одна из оставшихся вершин графа.

С первого взгляда кажется, что число областей из соотношения (5.23) может быть очень велико. Однако на практике пересечения множеств, входящие в выражение (5.23), становятся пустыми уже для небольших значений t. Это приводит к вполне обозримому множеству областей. Эффективные (с вычислительной точки зрения) методы построения областей, а также дальнейшего уменьшения числа необходимых областей приводятся в настоящем разделе несколько позже.

8.1. Описание алгоритма

Алгоритм построения абсолютного p-центра графа $G = (X, A)$ при заданном p выглядит следующим образом.

Шаг 1. Положить $\lambda = 0$.

Шаг 2. Увеличить λ на небольшую величину $\Delta\lambda$.

Шаг 3. Построить множества $Q_\lambda(x_i)$ для всех $x_i \in X_n$ и найти области Φ_λ.

Шаг 4. Образовать двудольный граф $G' = (X' \cup X, A')$, где X'—множество вершин, каждая из которых соответствует некоторой области Φ_λ, и A' — множество дуг, такое, что дуга между областью-вершиной и вершиной x_i существует тогда и только тогда, когда x_i может быть достигнута из этой области.

Шаг 5. Найти наименьшее доминирующее множество графа G' (см. гл. 3).

Шаг 6. Если число областей в приведенном выше множестве больше, чем p, то вернуться к шагу 2; в противном случае остановиться. Области этого множества образуют абсолютный p-центр исходного графа G.

Следует отметить (см. гл. 3), что число областей в наименьшем доминирующем множестве представляет (по определению) наименьшее число точек в G, из которых достигаются все вершины графа в пределах расстояния проникновения λ, используемого в данной итерации. Надо также отметить, что в процессе построения абсолютного p-центра по указанному выше алгоритму можно заодно получить абсолютные $(n-1)$-, $(n-2)$- и т. д. центры. Таким образом, если в конце некоторой итерации (продолжающейся с шага 2 до шага 6) число областей в наименьшем доминирующем

множестве уменьшается с некоторого уровня l до $l-1$, то области этого нового множества образуют абсолютный $(l-1)$-центр, а величина λ, взятая для этой итерации, будет «абсолютным $(l-1)$-радиусом», т. е. она является «критическим» значением λ — при меньших значениях λ не существуют $(l-1)$-центры, из которых в пределах взвешенного расстояния λ достижима каждая вершина графа.

Если нужно найти такое наименьшее значение p, что каждая вершина достижима из некоторого центра в пределах заданного критического расстояния (задача (b) из разд. 7), то шаги с 3 по 6 в приведенном выше алгоритме следует выполнять при λ, равном этому «критическому» значению. Соответствующее число областей в наименьшем доминирующем множестве является тогда требуемым значением для p, и области этого множества образуют искомый p-центр.

8.2. Вычислительные аспекты

Предположим, что λ зафиксировано и вычислены расстояния $\delta_i = \lambda/v_i$. Любое ребро графа либо достижимо целиком, либо частично, либо совсем не достижимо из вершины x_i в пределах расстояния δ_i. Если достижима только часть ребра (от какого-либо конца ребра до некоторой «предельной» точки на нем), то над предельной точкой ставится «метка». Эти метки содержат всю информацию, необходимую для описания множеств $Q_\lambda(x_i)$. Таким образом, $Q_\lambda(x_i)$ состоит из точек всех ребер (или частей ребер), принадлежащих кратчайшим маршрутам между метками и вершиной x_i.

После размещения всех меток (для всех вершин) каждое ребро будет разделено на ряд участков; каждый участок характеризуется теми вершинами, которые из него [1]) могут быть достигнуты. Таким образом, любой участок можно описать бинарным (единица-нуль) вектором $\{j_1, j_2, \ldots, j_n\}$ длины n, в котором $j_k = 1$, если вершина x_k достижима из этого участка, и $j_k = 0$ в противном случае.

Совокупность всех участков с одинаковым бинарным вектором образует область, и, следовательно, области Φ_λ, задаваемые соотношениями (5.23), могут быть построены с помощью таких меток. Поскольку область достижима только из тех вершин, для которых $j_k = 1$ (в бинарном векторном представлении этой области), и не из каких других, то в дальнейшем эти бинарные векторы мы будем называть *строгими пересечениями* (SI).

Представление области бинарным вектором не содержит никакой информации о месте ее расположения на графе. Однако такое представление выгодно с вычислительной точки зрения: оно не

[1]) То есть из точек этого участка.— *Прим. ред.*

предъявляет больших требований к памяти машины и позволяет уменьшить время вычисления, особенно при выполнении шага 5 алгоритма. Если абсолютный p-центр описан «на языке строгих пересечений», то дальнейшая процедура осуществляется чрезвычайно просто: надо лишь выяснить, какие участки ребер соответствуют этим строгим пересечениям.

На шаге 4 алгоритма надо строить двудольный граф $G' = (X' \cup X, A')$, у которого вершины из X' представляют области Φ_λ. Это может привести к графу больших размеров, что сильно увеличит время вычисления на шаге 5. Однако с помощью приводимой ниже теоремы удается уменьшить размеры графа G': исключаются те области, которые не влияют на получаемое оптимальное решение (но если существует больше одного оптимального решения, то при такой процедуре некоторые из них можно потерять). Другие пути, позволяющие достичь определенных упрощений, описаны в разд. 4.2 гл. 3.

Теорема 1. *При заданной величине λ для получения некоторого минимального доминирующего множества графа G' можно предварительно исключить из множества X' все вершины, соответствующие тем* SI, *над которыми доминируют другие* SI *в X'. Мы говорим, что* $(\mathrm{SI})_1$ *доминирует над* $(\mathrm{SI})_2$, *если* $(\mathrm{SI})_1 \otimes (\mathrm{SI})_2 = (\mathrm{SI})_2$, *где $\otimes$ означает булевское произведение.*

Доказательство этой теоремы очевидно.

В алгоритме из предыдущего раздела параметр λ при каждой очередной итерации увеличивается (по понятным причинам) лишь на небольшую величину. Существует много других более эффективных способов варьирования параметра λ. Результаты вычислений, приведенные в разд. 8.4, получены с помощью программы, использующей направленный двоичный поиск по λ.

8.3. Пример

Рассмотрим граф G, изображенный на рис. 5.8. Число, стоящее около любого ребра, задает его длину; вес каждой вершины графа равен единице. Требуется найти абсолютный p-центр с минимально возможной величиной параметра p и такой, чтобы каждая вершина графа отстояла хотя бы от одного из этих p центров на расстоянии, не превосходящем 3,5 единиц.

Для $\lambda = 3{,}5$ на рис. 5.9 показаны метки, определяющие множества $Q_\lambda(x_i)$. Они расставлялись сразу же в процессе последовательного «прохождения» ребер. Числа в кружках, стоящие около произвольной метки, указывают «номера» вершин, которым эта метка соответствует. Так, кружок ⑬ на ребре (2,3) означает, что

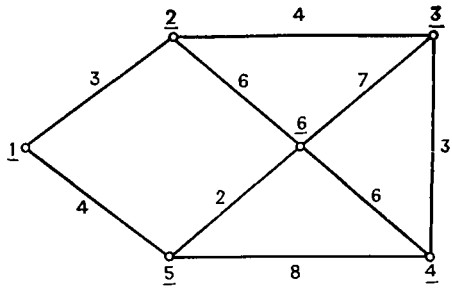


Рис. 5.8. Граф из примера 8.3.
Номера вершин подчеркнуты.
Все другие числа являются длинами ребер.

только отмеченная точка достижима из вершин 1 и 3 в пределах расстояния 3,5. На рис. 5.9 точками около ребер показано достижимое множество Q_λ (5), соответствующее вершине 5. Всего в данном примере существует 33 участка ребер, включая пустой участок (из которого ни одна вершина не может достигаться в пределах расстояния 3,5 единиц). Пустой участок расположен между метками 4 и 5 на ребре (4,5). Некоторые из этих 33 участков имеют

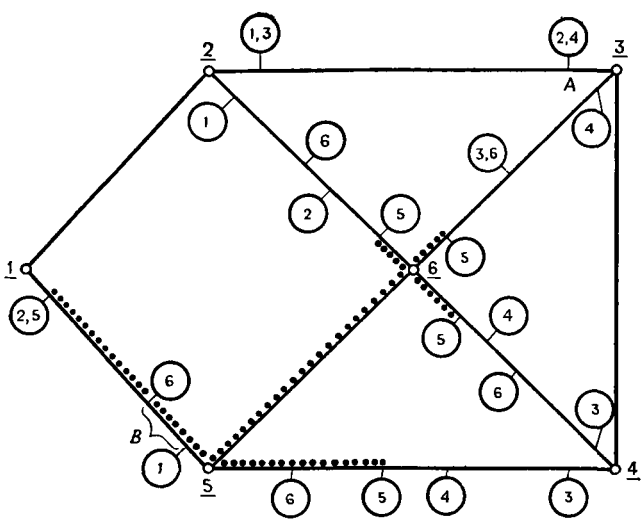


Рис. 5.9. Метки для достижимых множеств.
Множество Q_λ (5) отмечено точками.

одинаковые SI; существует всего 18 областей со следующими SI

(1) 000000	(7) 000011	(13) 000101
(2) 010000	(8) 110000	(14) 001001
(3) 001000	(9) 001100	(15) 111000
(4) 000100	(10) 100010	(16) 011100
(5) 000010	(11) 011000	(17) 110010
(6) 000001	(12) 010001	(18) 100011

Так, например, участок между метками 1 и 6 на ребре (1,5) принадлежит области с SI, равным 100011, поскольку лишь из точек графа, лежащих между этими двумя метками, можно достигнуть вершины 1, 5 и 6 (и никакие другие) в пределах расстояния $\lambda = 3{,}5$.

После исключения тех SI, которые доминируются другими SI, остается только 7 областей (с номерами SI от 12 до 18). Граф G'

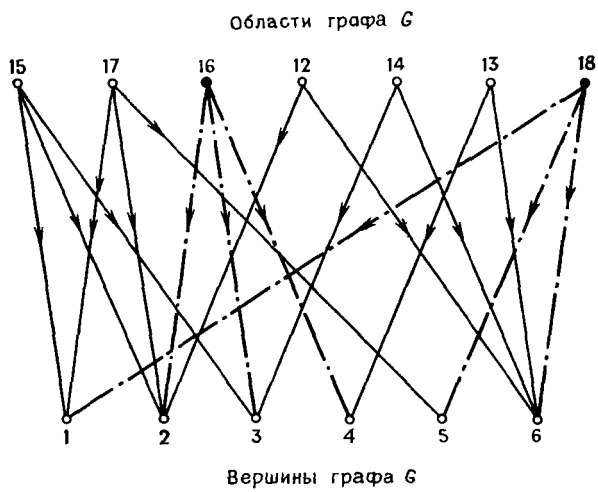


Рис. 5.10. Граф G' из примера 8.3.
● Области в наименьшем доминирующем множестве.

после выполнения шага 4 алгоритма выглядит так, как показано на рис. 5.10. Для нахождения наименьшего доминирующего множества этого графа можно воспользоваться алгоритмом, описанным в гл. 3 и предназначенным для решения задачи о наименьшем покрытии. Однако в нашем (простом) примере наименьшее множество легко строится с помощью прямого последова-

тельного перебора; оно «порождается» SI с номерами 16 и 18 в приведенном выше списке. Область, соответствующая SI с номером 16, состоит из одной точки, расположенной на ребре (2,3) в том месте, где находится кружок (2,4). На рис. 5.9 эта точка обозначена буквой A. Область, соответствующая SI с номером 18, представляет собой участок ребра (1,5) между метками 6 и 1; на рис. 5.9 она обозначена буквой B.

Таким образом, при $\lambda = 3{,}5$ требуется два центра; один расположен в точке A, а другой — в любой точке области B. Поскольку область A является точкой, то два указанных центра образуют также абсолютный 2-центр и $\lambda = 3{,}5$ является минимально возможным критическим расстоянием для существования 2-центра. Поэтому при $\lambda < 3{,}5$ область A исчезнет совсем, и тогда уже нужно строить 3-центр.

8.4. Результаты вычислений

Алгоритм построения абсолютного p-центра был опробован на ЭВМ CDC 6600. Каждый бинарный вектор, задающий SI, хранится в одном слове, так что можно использовать булевские функции для более удобного выполнения теоретико-множественных операций. Поскольку длина слова в вычислительной машине CDC 6600 равна 60 битам, то граф, содержащий не более 60 вершин, можно было непосредственно задавать в машинном коде.

В некоторых задачах, в частности, когда число областей в искомом абсолютном p-центре (т. е. число p) велико, шаг 5 алгоритма, описанного в разд. 8.1, требует значительную часть всего времени, затрачиваемого на решение задачи. В табл. 5.1 время, необходимое для выполнения этого шага, приведено отдельно от полного времени решения задачи. В этой таблице представлены времена вычисления (в секундах) и чи́сла итераций, требуемые при построении абсолютных 1-, 2-, 3-, 4- и 5-центров для некоторых графов. Здесь мы под числом итераций понимаем число повторений шагов алгоритма с 3 по 6 (для различных значений λ), всех тех повторений, которые необходимо осуществить для получения абсолютного p-центра с заданной точностью. Графы, которые были использованы при составлении табл. 5.1, выбирались случайным образом и являются связными и неориентированными. Требуемая точность во всех случаях принималась равной 1% от средней длины ребер рассматриваемого графа.

Как видно из табл. 5.1, алгоритм, приведенный в разд. 8.1, можно весьма эффективно использовать при нахождении абсолютных p-центров достаточно больших графов.

Таблица 5.1

Вычислительные результаты для выборочных графов

Граф		Число центров в абсолютном p-центре (значение p)														
		5			4			3			2			1		
n†	m‡	A	B	C	A	B	C	A	B	C	A	B	C	A	B	C
10	20	0,44	0,08	7	0,33	0,06	7	0,15	0,04	3	0,30	0,04	6	0,34	—	7
10	30	0,46	0,06	6	0,35	0,05	6	0,46	0,09	7	0,55	0,08	7	0,51	—	7
20	30	14,00	13,00	8	1,36	0,45	7	1,16	0,11	8	1,13	0,05	8	0,86	—	6
20	40	3,51	1,94	7	0,37	0,22	1	1,49	0,43	7	1,73	0,06	6	1,05	—	5
20	60	6,02	4,50	7	2,58	1,06	7	1,29	0,18	5	1,63	0,05	6	1,75	—	7
20	80	16,10	12,00	9	3,60	1,70	8	2,70	0,41	6	3,20	0,10	8	2,90	—	8
20	100	—	—	—	4,25	1,53	6	2,85	0,37	6	5,41	0,19	9	2,20	—	4
20	120	—	—	—	6,38	2,86	6	4,21	0,58	7	4,35	0,09	6	3,32	—	4
30	40	—	—	—	6,00	2,10	11	3,82	0,45	10	3,83	0,05	10	5,00	—	12
30	60	—	—	—	18,10	14,50	7	2,66	0,12	5	3,16	0,06	6	2,22	—	4
30	80	—	—	—	8,40	3,73	6	7,20	1,56	9	6,90	0,17	10	3,55	—	5
30	100	—	—	—	23,20	18,80	6	6,50	1,20	6	7,10	0,25	8	6,60	—	8
40	60	—	—	—	—	—	—	11,80	2,53	10	11,20	0,22	13	12,40	—	14
40	80	—	—	—	—	—	—	27,00	13,50	11	10,00	0,51	8	7,40	—	6
50	80	31,34	25,88	12	24,55	16,60	13	17,70	0,95	12	8,11	0,11	6	17,80	—	12

† *Число вершин*
‡ *Число ребер*
A *Полное время вычисления (секунды на CDC 6800).*
B *Время вычисления для шага 5 алгоритма*
C *Число итераций*

8.5. Применение общего алгоритма для поиска p-центров

Алгоритм из разд. 8.1 можно, очевидно, использовать при решении более узкой задачи, приведенной в разд. 6, т. е. при нахождении p-центра. Для этого надо только изменить шаг 6 настоящего алгоритма (см. разд. 8.1) таким образом, чтобы «контролировалось» не *число* областей в наименьшем доминирующем множестве, найденном на шаге 5, а *число вершин графа* G, *содержащихся* в этих областях. Если полученное число вершин больше p, то нужно возвратиться к шагу 2. В противном случае алгоритм заканчивает работу, и p-центром графа будет множество вершин, содержащихся во всех тех областях, которые образуют доминирующее множество.

Поскольку в этом случае необходимы только вершины, содержащиеся в областях, задаваемых соотношениями (5.22) и (5.23), то алгоритм из разд. 8.1 можно без изменений применять для нахождения p-центров, если области на шаге 3 строить с помощью соотношений (5.22) и (5.23), используя вместо множеств $Q_\lambda(x_i)$ множества $R_\lambda(x_i)$. Более того, если все $n(n-1)$ расстояний в матрице расстояний с самого начала расположить в строку в порядке неубывания $[f_1, f_2, \ldots, f_{n(n-1)}]$, то на шаге 2 алгоритма значения для параметра λ должны выбираться лишь из этой строки, и их не надо увеличивать на произвольно малую величину в каждой итерации. Поиск начинается с $\lambda = f_1$, а затем продолжается при $\lambda = f_2$, f_3 и т. д. в соответствии с указанным выше упорядочением. Следует заметить, что здесь, очевидно, было бы весьма кстати бинарное представление семейства [1]) $\{f_1, f_2, \ldots, f_{n(n-1)}\}$.

9. Задачи

1. Найти центр графа, приведенного на рис. 5.11. Рядом с дугами указаны их стоимости, а веса вершин задаются вектором [5, 3. 1, 7, 4, 6]

2. Найти абсолютные центр и радиус неориентированного графа, изображенного на рис. 5.12, предполагая, что веса всех вершин равны 1. Использовать метод Хакими, а также итерационный метод из разд. 5.4. Сравнить вычислительную трудоемкость этих двух методов.

3. Показать, что в любом неориентированном графе G с единичными весами вершин существует такой маршрут между неко-

[1] В *семействе* (в отличие от *множества*) могут встречаться одинаковые элементы.— *Прим. ред.*

торой парой вершин x_i и x_j, что абсолютный центр y^* графа G лежит в середине этого маршрута.

4. Используя свойство, приведенное выше в задаче 3, найти абсолютный центр в графе, изображенном на рис. 5.12.

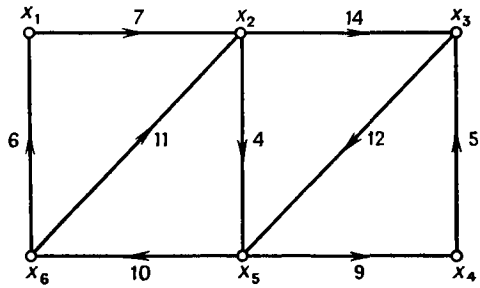


Рис. 5.11.

5. Определить понятие «середина пути» и показать, что свойство, приведенное выше в задаче 3, может быть обобщено на неориентированные графы с произвольными весами вершин $[v_j]$.

6. Используя алгоритм из разд. 8, найти абсолютный 3-центр графа G, приведенного на рис. 5.12, и показать, что каждая вершина графа G достижима по крайней мере из одной области абсолютного 3-центра в пределах расстояния $\lambda = 6$. Каково наименьшее значение параметра p, обеспечивающее достижимость всех вершин графа G из абсолютного p-центра в пределах расстояния $\lambda = 5{,}5$?

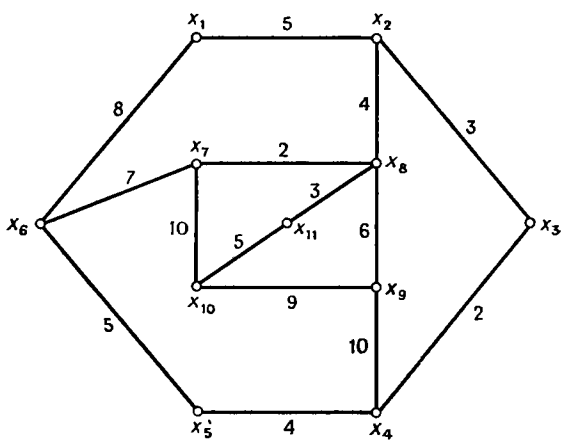


Рис. 5.12.

7. Используя алгоритм из разд. 8.5, найти 3-центр графа, изображенного на рис. 5.12, и показать, что $\lambda = 8$ является наименьшим возможным λ, определяющим существование 3-центра.

8. Пусть X_p^* — p-центр в неориентированном графе G. Доказать утверждение: ребрам графа G можно придать такую ориентацию, что X_p^* будет внешним p-центром в новом, ориентированном графе и расстояния $d(X_p^*, x_i)$ для всех вершин x_i останутся неизменными.

9. Показать, что если G — связный неориентированный граф с единичными стоимостями ребер, то наименьшее значение параметра p, обеспечивающее существование p-центра с достижимостью, ограниченной числом λ, удовлетворяет неравенству

$$p \leqslant \frac{n}{[\lambda]+1},$$

где $[\lambda]$ — целая часть λ (см. [5]). Предполагается, что $\lambda < n$.

10. Используя результат задачи 9, показать, что, каково бы ни было целое положительное δ, все n вершин связного неориентированного графа G можно покрыть $\frac{n}{[\delta/2]+1}$ подграфами графа G, имеющими диаметры, не превосходящие δ (см. разд. 5.4 и работу [5]).

11. Показать, что для связного неориентированного графа G радиус ρ и диаметр δ удовлетворяют соотношению $\delta/2 \leqslant \rho \leqslant \delta$. Привести примеры графов, для которых обе оценки радиуса являются достижимыми.

12. Показать, что в случае ориентированного графа G неравенства, подобные приведенным в задаче 11, не выполняются.

13. Заново определим число внешнего разделения $s_0(y)$ и внешний абсолютный центр y_0^* графа G:

$$s_0(y) = \max_{y' \text{ на } G} [d(y, y')],$$

y_0^* является такой точкой y графа G, для которой

$$s_0(y_0^*) = \min_{y \text{ на } G} [s_0(y)].$$

Обобщить методы, приведенные в разд. 5, таким образом, чтобы можно было строить определенные выше внешние абсолютные центры графов.

10. Список литературы

1. Bollobas B. (1968), Graphs of given diameter, Theory of Graphs, Erdos and Katona, Eds., Academic Press, New York.
2. Bosak J., Rosa A., Znam Š. (1968), On decomposition of complete graphs into factors with given diameters, Theory of Graphs, Erdös and Katona, Eds., Academic Press, New York.
3. Christofides N., Viola P. (1971), The optimum location of multi-centers on a graph, *Opl. Res. Quart.*, **22**, p. 45.
4. Dijkstra D. (1959), A note an two problems in connection with graphs, *Numerische Mathematik*, **1**, p. 269.
5. Erdös P., Gerencsér L., Máté A. (1969), Problems of graph theory concerning optimal design, Colloquia Mathematica Societatis Janos Bolyai, № 4, Combinatorial theory and its applications, Balatonfured (Hungary), p. 317.
6. Frank H., Frisch I. T. (1971), Communication transmission and transportation networks, Addison Wesley, Reading, Massachusetts.
7. Hakimi S. L. (1964), Optimum location of switching centres and the absolute centres and medians of a graph, *Ops. Res.*, **12**, p. 450.
8. Hakimi S. L. (1965), Optimum distribution of switching centres in a communication network and some related graph theoretic problems, *Ops. Res.*, **13**, p. 462.
9. Minieka E. (1970), The *m*-centre problem, *J. of SIAM* (review), **12**, p. 138.
10. Rosenthal M. R., Smith S. B. (1967), The *m*-centre problem, Presented at the 31st National ORSA Meeting, New York.
11. Sakowicz A. F. (1970), A planning model for the location of communication centres, Presented at the 38th National ORSA Meeting, Detroit, Michigan.
12. Singer S. (1968), Multi-centres and multi-medians of a graph with an application to optimal warehouse location, Presented at the 16th TIMS Conf.
13. Toregas C., Swain R., Revelle C., Bergman L. (1971), The location of emergency service facilities, *Ops. Res* , **19**, p. 1363.

Глава 6

РАЗМЕЩЕНИЕ МЕДИАН В ГРАФЕ

1. Введение

В ряде задач о размещении пунктов обслуживания требуется так расположить пункт обслуживания на графе, чтобы сумма кратчайших расстояний от этого пункта до вершин графа была минимально возможной. Оптимальное в указанном смысле место расположения пункта называется *медианой* графа. Исходя из природы целевой функции, такие задачи называют *минисуммными задачами размещения*. Эти задачи в различных формах часто встречаются на практике: при выборе места расположения коммутаторов в телефонной сети, подстанций в электросетях, баз снабжения (складов) в сети дорог (где вершины представляют потребителей) или отделов сортировки в почтовой связи.

С этим типом задач контрастирует *минимаксная задача размещения*, возникающая при выборе места для таких пунктов обслуживания, как пожарное депо, полицейский участок или амбулатория, и рассмотренная в предыдущей главе. В настоящей главе рассматривается минисуммная задача размещения. В частности, обсуждается задача о нахождении *p-медианы* данного графа G; это задача об оптимальном размещении заданного числа (скажем, p) пунктов обслуживания, при котором сумма кратчайших расстояний от вершин графа G до ближайших к ним пунктов [1]) принимает минимально возможное значение. Задача нахождения p-медианы может быть несколько обобщена, если каждой вершине x_j сопоставить некоторый вес v_j (представляющий, например, ее размеры или важность); тогда целевой функцией, подлежащей минимизации, станет сумма «взвешенных» расстояний.

2. Медиана графа

Пусть дан граф $G = (X, \Gamma)$. Для каждой вершины $x_i \in X$ определим два числа, которые назовем *передаточными* числами:

$$\left.\begin{aligned} \sigma_0(x_i) &= \sum_{x_j \in X} v_j d(x_i, x_j), \\ \sigma_1(x_i) &= \sum_{x_j \in X} v_j d(x_j, x_i), \end{aligned}\right\} \tag{6.1}$$

[1]) Суммирование ведется по всем вершинам. Для каждой вершины берется кратчайшее расстояние от нее до ближайшего к ней пункта обслуживания.— *Прим. ред.*

где $d(x_i, x_j)$ — кратчайшее расстояние от вершины x_i до вершины x_j. Числа $\sigma_o(x_i)$ и $\sigma_t(x_i)$ называются соответственно *внешним* и *внутренним передаточными числами* вершины x_i. Число $\sigma_o(x_i)$ есть сумма элементов строки x_i матрицы, полученной после умножения каждого столбца матрицы расстояний $D(G) = [d(x_i, x_j)]$ на вес соответствующей этому столбцу вершины, т. е. j-й столбец умножается на v_j; число $\sigma_t(x_i)$ есть сумма элементов столбца x_i матрицы, полученной в результате умножения каждой строки матрицы расстояний $D(G)$ на соответствующий этой строке вес (j-я строка умножается на v_j).

Вершина $\bar{x}_o$, для которой

$$\sigma_o(\bar{x}_o) = \min_{x_i \in X} [\sigma_o(x_i)], \tag{6.2}$$

называется *внешней медианой* графа G, а вершина $\bar{x}_t$, для которой

$$\sigma_t(\bar{x}_t) = \min_{x_i \in X} [\sigma_t(x_i)], \tag{6.3}$$

называется *внутренней медианой* графа G.

Рассмотрим еще раз граф, изображенный на рис. 5.1 (для которого все v_j и c_{ij} приняты равными единице), и вычислим внешние и внутренние передаточные числа вершин. Эти числа приведены в присоединенных к матрице расстояний строке и столбце:

$D(G) =$

	x_1	x_2	x_3	x_4	x_5	x_6	$\sigma_o(x_i)$ ↓
x_1	0	1	2	3	2	3	11
x_2	3	0	1	2	1	2	9
x_3	4	3	0	1	2	3	13
x_4	3	2	2	0	1	2	10
x_5	2	1	1	2	0	1	7*
x_6	1	2	3	4	3	0	13
$\sigma_t(x_i) \rightarrow$	13	9*	9*	12	9*	11	

По полученным передаточным числам видно, что внешней медианой является x_5 с $\sigma_o(x_5) = 7$ и что существуют три внутренние медианы (x_2, x_3 и x_5), каждая с внутренним передаточным числом, равным 9.

2.1. Выбор места для склада

Рассмотрим задачу снабжения ряда потребителей товарами поступающими с одного склада. Потребителей можно объединить в группы таким образом, чтобы каждую группу обслуживало *целое число грузовиков*. Машина выезжает со склада, обслуживает некоторую группу потребителей и возвращается на склад. Группы потребителей можно представить вершинами графа, а сеть дорог — его дугами. На практике каждой группе потребителей присваивается вес v_j, представляющий ее «важность» (например, v_j может быть числом, пропорциональным годовому потреблению или частоте, с которой транспорт должен объезжать эту группу потребителей, чтобы удовлетворить их потребности).

В этом случае задача состоит в определении такого места для склада, чтобы общее расстояние, проходимое транспортом, было бы минимально возможным. Если матрица расстояний $D(G)$ задает действительные расстояния в километрах, то требуемым местом расположения склада будет такая вершина $\bar{x}_{o,t}$, для которой сумма внешних и внутренних передаточных чисел будет наименьшей. Вершина $\bar{x}_{o,t}$ может быть названа *внешне-внутренней медианой* и найдена из соотношения, аналогичного (6.1). В следующем разделе будет показано, что для любой точки (вершины или произвольной точки дороги), выбранной для размещения склада, общий километраж, покрываемый транспортом, не может быть меньше, чем для вершины $\bar{x}_{o,t}$.

3. Кратные медианы (p-медианы) графа

В предыдущей главе, обобщая понятие центра, мы ввели понятие p-центра. Подобным же образом можно обобщить понятие медианы, определив p-медиану.

Пусть X_p — подмножество множества вершин X графа $G = (X, \Gamma)$, и предположим, что X_p содержит p вершин. Как и прежде, введем следующие обозначения:

$$d(X_p, x_j) = \min_{x_i \in X_p} [d(x_i, x_j)] \qquad (6.4\,(а))$$

и

$$d(x_j, X_p) = \min_{x_i \in X_p} [d(x_j, x_i)]. \qquad (6.4\,(б))$$

Если x_i' — вершина из X_p, на которой достигается минимум в (6.4 (а)) или (6.4 (б)), то говорят, что вершина x_j *прикреплена к* x_i'. Передаточные числа множества вершин X_p определяются

так же, как и для одиночной вершины:

$$\left.\begin{aligned}\sigma_o(X_p) &= \sum_{x_j \in X} v_j d(X_p, x_j),\\ \sigma_t(X_p) &= \sum_{x_j \in X} v_j d(x_j, X_p),\end{aligned}\right\} \tag{6.5}$$

где $\sigma_o(X_p)$ и $\sigma_t(X_p)$ — соответственно внешнее и внутреннее передаточные числа множества вершин X_p.

Множество $\overline{X}_{po}$, для которого

$$\sigma_o(\overline{X}_{po}) = \min_{X_p \subseteq X} [\sigma_o(X_p)], \tag{6.6}$$

называют *внешней p-медианой* графа G; аналогично определяется *внутренняя p-медиана*.

Как и в случае p-центров, рассмотренных в предыдущей главе, даже для графов средних размеров с вычислительной точки зрения нецелесообразно использовать при нахождении p-медиан непосредственно выражения (6.4), (6.5) и (6.6). Алгоритмы построения p-медиан будут даны в разд. 5.

3.1. Абсолютные p-медианы

С целью упрощения изложения рассмотрим неориентированный граф G. Индексы o и t будут отсутствовать. Разберем сначала случай медианы (1-медианы). Спрашивается, существует ли такая точка y на некотором ребре (не обязательно совпадающая с вершиной графа G), для которой передаточное число

$$\sigma(y) = \sum_{x_j \in X} v_j d(y, x_j)$$

меньше, чем для медианы графа G. Точку $\bar{y}$, на которой достигается минимум величины $\sigma(y)$, будем называть *абсолютной медианой* графа G.

Сейчас мы докажем [5,6], что не существует точки $\bar{y}$, для которой $\sigma(\bar{y}) < \sigma(\bar{x})$, т. е. здесь ситуация противоположна той, которая имела место при рассмотрении центров.

Теорема 1. *Какова бы ни была точка y графа $G = (X, A)$, в нем найдется по крайней мере одна вершина x, для которой* $\sigma(x) \leqslant \sigma(y)$.

Доказательство. Пусть y — точка ребра (x_a, x_b), расположенная на расстоянии ξ от x_a. Тогда

$$d(y, x_j) = \min [\xi + d(x_a, x_j), c_{ab} - \xi + d(x_b, x_j)], \tag{6.7}$$

где c_{ab} — длина ребра (x_a, x_b).

Пусть X_a — множество тех вершин x_j, для которых первый член в (6.7) не больше второго, а X_b — множество вершин, для которых второй член меньше первого. Мы можем тогда написать

$$\sigma(y) = \sum_{x_j \in X} v_j d(y, x_j) = \sum_{x_j \in X_a} v_j [\xi + d(x_a, x_j)] + \\ + \sum_{x_j \in X_b} v_j [c_{ab} - \xi + d(x_b, x_j)]. \quad (6.8)$$

Поскольку из неравенства треугольника следует, что

$$d(x_a, x_j) \leqslant c_{ab} + d(x_b, x_j), \quad (6.9)$$

то, заменяя $c_{ab} + d(x_b, x_j)$ на $d(x_a, x_j)$ в выражении (6.8), получаем

$$\sigma(y) \geqslant \sum_{x_j \in X_a} v_j [\xi + d(x_a, x_j)] + \sum_{x_j \in X_b} v_j [d(x_a, x_j) - \xi]. \quad (6.10)$$

Так как $X_a \cup X_b = X$, то, сделав перегруппировку в (6.10), имеем:

$$\sigma(y) \geqslant \sum_{x_j \in X} v_j d(x_a, x_j) + \xi \left[\sum_{x_j \in X_a} v_j - \sum_{x_j \in X_b} v_j \right]. \quad (6.11)$$

Поскольку для каждого ребра (x_a, x_b) мы вправе сами решать, какую вершину называть x_a и какую x_b, то всегда можно добиться выполнения неравенства

$$\sum_{x_j \in X_a} v_j \geqslant \sum_{x_j \in X_b} v_j.$$

Заметив, что первый член в правой части неравенства (6.11) равен $\sigma(x_a)$, получаем из (6.11) такое соотношение:

$$\sigma(y) \geqslant \sigma(x_a).$$

Таким образом, для вершины x_a величина $\sigma(x_a)$ не превышает $\sigma(y)$ и, следовательно, теорема доказана.

Теорема 1 довольно просто обобщается на случай p-медиан.

Теорема 2. *Каково бы ни было множество Y_p, состоящее из p точек графа $G = (X, A)$, т. е. из p точек ребер и вершин, найдется по крайней мере одно подмножество $X_p \subset X$, содержащее p вершин, для которого $\sigma(X_p) \leqslant \sigma(Y_p)$.*

В теоремах 1 и 2 предполагалось, что передаточные числа $\sigma(x)$ и $\sigma(X_p)$ определены с помощью выражений (6.1) и (6.5). В работах Леви [20], Голдмана [12] и Голдмана и Мейерса [14] было показано, что эти теоремы остаются в силе и в тех случаях, когда передаточные числа определяются как суммы произвольных вогнутых функций от взвешенных расстояний.

Из теорем 1 и 2 следует, что понятие абсолютной медианы не представляет особого интереса (в противоположность ситуации с абсолютными центрами, рассмотренными в гл. 5). Поэтому в остальной части этой главы основное внимание уделяется задаче о p-медиане.

4. Обобщенная p медиана графа

Задача нахождения p-медианы графа является центральной в общем классе задач, встречающихся в литературе под названием «распределение и размещение центров обслуживания» [5, 6, 9] или «размещение складов» [8, 29, 30, 10, 15, 7, 28]. Эти задачи являются до некоторой степени более общими, чем задача о p-медиане, в которой вершинам сопоставлены фиксированные стоимости f_i. *Обобщенная задача о p-медиане* может быть сформулирована следующим образом.

Для заданного графа $G = (X, A)$ с матрицей кратчайших расстояний $[d(x_i, x_j)]$, с весами вершин v_i и с фиксированными стоимостями вершин f_i задача состоит в нахождении такого подмножества $\overline{X}_p$ из p вершин, для которого величина

$$z = \sum_{x_i \in \overline{X}_p} f_i + \sigma(\overline{X}_p) \tag{6.12}$$

принимает минимально возможное значение.

Таким образом, в этом случае минимизации подлежит не просто передаточное число $\sigma(\overline{X}_p)$ множества $\overline{X}_p$, а полная функция цели, содержащая фиксированные стоимости f_i вершин x_i подмножества $\overline{X}_p$. На практике f_i представляет, например, фиксированную стоимость строительства пункта обслуживания (склада, фабрики, подстанции) при размещении его в вершине x_i. Задача о p-медиане соответствует такому случаю, когда все f_i одинаковы (например, равны f), так что первый член в (6.12) равен постоянной величине pf независимо от выбора подмножества $\overline{X}_p$

Очень близкой к сформулированной выше задаче об обобщенной p-медиане является такая задача, в которой число вершин $|\overline{X}_p|$ не обязательно равно p, а может быть некоторым числом, не превосходящим p. Задача минимизации выражения (6.12) при условии $|\overline{X}_p| \leqslant p$ является разновидностью задачи об обобщенной p-медиане и часто встречается на практике.

В задачах размещения складов неизменно приходится сталкиваться с ограничениями, которых нет в «чистой» задаче о p-медиане. Чаще всего встречаются ограничения на наибольшее и наименьшее значения, которые может принять выражение

$$\sum_{x_j, \text{ прикрепленные к } x_i} v_j \tag{6.13}$$

для любой медианной вершины $x_i \in \overline{X}_p$. Выражение (6.13) определяет количество товара, вывозимого из вершины x_i, и, следовательно, является мерой вместимости склада x_i [8, 15, 7, 28].

Очевидно, что такое же обобщение, возникающее при введении ограничений на величину (6.13), возможно и в задачах о нахождении p-центров и абсолютных p-центров, рассмотренных в предыдущей главе. Однако для практических задач размещения энергетических объектов, именно тех задач, которые связаны с нахождением p-центров, указанное обобщение имеет небольшое значение, так как всегда эти ограничения легко вводятся в алгоритм, приведенный в разд. 8 предыдущей главы.

Сложность решения практических задач размещения складов определяется не изменениями и дополнительными ограничениями, рассмотренными выше, а свойственна природе самой задачи о p-медиане. В свете этого рассмотрим несколько методов, пригодных для определения p-медианы графа. Для упрощения изложения индексы o и t, указывающие связь объектов соответственно с внешними и внутренними медианами, будут опускаться, поскольку излагаемые методы применимы к обеим задачам.

5. Методы решения задачи о p-медиане

5.1. Формулировка задачи в терминах целочисленного программирования

Пусть $[\xi_{ij}]$ — матрица распределения, в которой

$$\xi_{ij} = \begin{cases} 1, & \text{если вершина } x_j \text{ прикреплена к вершине } x_i, \\ 0 & \text{в противном случае.} \end{cases}$$

Далее примем $\xi_{ii} = 1$, если вершина x_i является медианной вершиной и $\xi_{ii} = 0$ в противном случае. Задача о p-медиане может быть сформулирована тогда следующим образом.

Минимизировать функцию

$$z = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} \xi_{ij} \tag{6.14}$$

при ограничениях

$$\sum_{i=1}^{n} \xi_{ij} = 1 \qquad \text{для} \quad j = 1, \ldots, n, \tag{6.15}$$

$$\sum_{i=1}^{n} \xi_{ii} = p, \tag{6.16}$$

$$\xi_{ij} \leqslant \xi_{ii} \qquad \text{для всех } i, j = 1, \ldots, n \tag{6.17}$$

и

$$\xi_{ij}=0 \quad \text{или} \quad 1, \tag{6.18}$$

где $[d_{ij}]$ — матрица взвешенных расстояний графа (она получается из матрицы расстояний после умножения каждого j-го столбца на соответствующий вес v_j). Соотношения (6.15) гарантируют выполнимость следующего условия: любая заданная вершина x_j прикреплена к одной и только к одной медианной вершине x_i. Выражение (6.16) гарантирует существование в точности p медианных вершин. Из ограничений (6.17) следует, что $\xi_{ij}=1$ только тогда, когда $\xi_{ii}=1$, т. е. прикрепления осуществляются только к вершинам медианного множества. Если $[\bar{\xi}_{ij}]$ является оптимальным решением задачи, определяемой соотношениями (6.14)—(6.18), то p-медиана имеет вид

$$\bar{X}_p=\{x_i \mid \bar{\xi}_{ii}=1\}.$$

Если ограничения (6.18) вышеприведенной задачи заменить на

$$\xi_{ij}\geqslant 0 \tag{6.19}$$

то возникает задача линейного программирования (ЛП). Ее решение получить нетрудно. (Отметим, что для величины ξ_{ij} указывать верхнюю границу не нужно, поскольку из соотношений (6.15) следует, что $\xi_{ij}\leqslant 1$.)

Решение задачи ЛП не обязательно будет целочисленным, ξ_{ij} может принимать и дробные значения. Ревель и Свэйн [26] показали, что дробные значения встречаются крайне редко, и поэтому в большинстве случаев для получения p-медиан можно использовать язык и методы линейного программирования. В том случае, когда некоторые ξ_{ij} являются дробными, решение можно получить с помощью древовидного [1]) поиска [11], причем одной ветви, соответствующей какой-либо дробной величине (переменной) ξ_{ij}, придается значение 0, а другой ветви — значение 1. Затем для каждой из полученных ветвей (подзадач) можно продолжить поиск решения (как задачи ЛП) и действовать подобным образом до тех пор, пока все ξ_{ij} не примут значения из множества $\{0, 1\}$.

Другой подход, отличный от метода линейного программирования, предложен Марстеном [23], который показал, что решение $[\bar{\xi}_{ij}]$ задачи (6.14)—(6.18), соответствующее p-медиане графа, является экстремальной точкой некоторого многогранника H, причем многогранник будет одним и тем же для всех p, удовлетворяющих условию $1\leqslant p\leqslant n$. Используя множители Лагранжа и параметрическое линейное программирование, Марстен предло-

[1]) Имеется в виду «поиск, использующий дерево решений» (в оригинале — tree search).— *Прим. ред.*

жил метод «прокладки пути» между несколькими экстремальными точками многогранника H. При «прокладке» этого пути последовательно, в порядке убывания p, порождаются некоторые p-медианы графа. Кое-какие p-медианы (для отдельных значений p) могут быть не построены, и, наоборот, могут быть «порождены» такие экстремальные точки многогранника H, которые не соответствуют p-медианам графа G, так как содержат дробные значения величин ξ_{ij}. Хотя этот метод привлекателен как с теоретической, так и с вычислительной точек зрения, но все же он не всегда приводит к построению p-медианы графа для требуемого значения p. В работе [23] дан пример полного графа с 33 вершинами, для которого указанным методом последовательно порождаются p-медианы с $p = 33, 32, \ldots, 10$, но построить таким способом 9-медиану и 8-медиану уже нельзя.

5.2. Алгоритм направленного древовидного поиска

Вместо того чтобы задачу нахождения p-медианы толковать как задачу целочисленного программирования, можно для ее решения использовать прямое дерево поиска. Этот подход лучше всего соответствует структуре рассматриваемой задачи. В настоящем разделе будет описан один из таких методов, в котором каждая подзадача, возникающая при ветвлении в каком-либо узле дерева, определяется тем, что (при заданной вершине x_j) переменная ξ_{ij} полагается равной 1 для некоторой вершины x_i. Равенство $\xi_{ij} = 1$ означает, что вершина x_j прикреплена к вершине x_i, а также, очевидно, что x_i является медианной вершиной.

Этот поиск удобно выполнять следующим образом.

Построим матрицу $M = [m_{kj}]$, j-й столбец которой содержит все вершины графа G, расположенные в порядке неубывания их расстояний от вершины x_j. Таким образом, если $m_{kj} = x_i$, то вершина x_i будет k-й ближайшей к x_j вершиной. Очевидно, что первой ближайшей к x_j вершиной является она сама, т. е. $m_{1j} = x_j$.

Поиск начинается с последовательного просмотра всех вершин графа, с x_1 по x_n. Вершина x_j вначале прикрепляется к вершине m_{1j}, затем к m_{2j} и т. д., пока не будут перебраны все возможности. Необходимо сделать несколько замечаний.

1. Поскольку оптимальное решение состоит из p медианных вершин, то прикрепление вершины в этом решении к какой-либо медианной вершине должно быть наилучшим из p возможных прикреплений. Иными словами, для каждой вершины должно существовать по крайней мере еще $p - 1$ возможностей прикрепления с неменьшей стоимостью, чем у выбранной возможности. Следовательно, из матрицы M можно удалить $p - 1$ последних строк, и это не отразится на оптимальном решении задачи о p-медиане.

2. Предположим, что вершину x_j прикрепили к вершине $m_{k'j'}$ (которая является, например, вершиной x_i). Пусть вершина x_i будет k-й ближайшей вершиной к некоторой еще не прикрепленной вершине x_j ($j' < j \leqslant n$), соответствующей j-му столбцу матрицы M, т. е. $m_{kj} = x_i$. Тогда, очевидно, все элементы m_{lj} с $l > k$ могут быть исключены из дальнейшего рассмотрения (отмечены), поскольку, прикрепив $x_{j'}$ к x_i, мы тем самым причисляем x_i к медианным вершинам и поскольку вершина x_j может быть прикреплена к x_i с меньшей стоимостью [1]), чем к любой другой вершине m_{lj} при $l = k + 1, k + 2, \ldots$. Ясно также, что если на некотором шаге возвращения процедуры поиска изменится распределение вершины $x_{j'}$ относительно x_i, то элементы m_{lj} (неисключенные из рассмотрения, неотмеченные) должны быть рассмотрены заново.

3. Пусть вершина $x_{j'}$ прикреплена к вершине $m_{k'j'}$. Тогда можно предположить, что все вершины $m_{1j'}, m_{2j'}, \ldots, m_{(k'-1)j'}$ не являются медианными вершинами, поскольку в противном случае вершина $x_{j'}$ была бы расположена по крайней мере от одной из них не дальше, чем от $m_{k'j'}$ (с меньшей [1]) результирующей стоимостью). Эти вершины, следовательно, могут быть отмечены (исключены из рассмотрения) во всех столбцах с номерами $j > j'$. Нужно иметь в виду, что вершины эти исключаются временно и их надо «восстанавливать» всякий раз при изменении распределения вершины $x_{j'}$ относительно $m_{k'j'}$.

4. Если из рассмотрения исключены t верхних элементов j-го столбца, соответствующего нераспределенной (неприкрепленной) вершине x_j, и $(t + 1)$-й элемент, т. е. $m_{(t+1)\,j}$ является медианной вершиной, то x_j должна быть прикреплена к вершине $m_{(t+1)j}$. В этом случае нет необходимости рассматривать какие-либо другие возможности до тех пор, пока некоторые из t верхних элементов не окажутся «восстановленными» (вследствие изменения в предыдущих распределениях вершин, что отразится на множестве «исключенных» вершин). Этот вывод является прямым следствием вышеприведенных замечаний 2 и 3.

5. Если на некотором этапе q проводимого поиска будет построено множество из p медианных вершин (в результате применения «процедуры прикрепления» вершин), то каждую оставшуюся нераспределенную вершину можно прикрепить к ближайшей медианной вершине. Очевидно, что это является оптимальным завершением частного решения, соответствующего осуществленному прикреплению вершин. На следующем шаге процедуры — шаге возвращения — должно изменяться распределение вершин, полученное в конце q-го этапа.

---

[1]) Точнее, с не большей стоимостью.— *Прим. ред.*

5.2.1. Вычисление нижней границы. Замечания 1—5 можно использовать для ограничения размера дерева поиска путем уменьшения числа возможных прикреплений вершины x_j на любом этапе процедуры. Кроме того, на некоторых этапах, когда последней распределяемой вершиной является вершина $x_{j'}$, для ограничения дальнейшего поиска может быть использована нижняя граница величины полного оптимального решения (полученного для уже осуществленных прикреплений).

Предположим, что выполненные прикрепления (включая прикрепление вершины $x_{j'}$) дали p' ($p' < p$) медианных вершин. Тогда оставшиеся прикрепления должны привести к нахождению остальных $p - p'$ медианных вершин. Пусть J — множество индексов еще не распределенных вершин. В общем случае J есть множество индексов j, для которых $j' < j \leqslant n$, за исключением индексов тех вершин, распределение которых индуцировано распределением первых j' вершин $x_1, x_2, \ldots, x_{j'}$ (в соответствии с замечанием 4).

Пусть $m_{\alpha_j j}$ и $m_{\beta_j j}$ — самый верхний и следующий за ним элементы j-го столбца, которые являются неотмеченными (не исключенными из рассмотрения). Тогда наилучшим распределением вершины x_j будет ее прикрепление к вершине $m_{\alpha_j j}$. Если число различных вершин $m_{\alpha_j j}$ для $j \in J$ равно h и $h = p - p'$, то все эти наилучшие распределения еще не прикрепленных вершин являются осуществимыми (т. е. получается множество, содержащее p медианных вершин). Эти распределения образуют оптимальное пополнение частного решения, полученного при распределении вершин $x_1, x_2, \ldots, x_{j'}$. В таком случае записывается результат и шаг возвращения может быть предпринят с текущего частного решения.

Если, однако, $h > p - p'$, то для получения всех p медианных вершин надо заменить по крайней мере $h - p + p'$ лучших назначений на вторые лучшие или худшие. Тогда наименьшая дополнительная стоимость распределения является суммой $h - p + p'$ наименьших разностей

$$v_j [d(x_j, m_{\beta_j j}) - d(x_j, m_{\alpha_j j})] \tag{6.20}$$

по всем вершинам x_j, $j \in J$.

Нижняя граница стоимости оптимального решения, даваемая текущим частным решением, получается прибавлением к сумме стоимостей уже выполненных распределений

$$\sum_{j \notin J} v_j d(x_j, m_{\alpha_j j})$$

суммы $h - p + p'$ наименьших разностей, задаваемых выражением (6.20).

Может случиться, что h меньше $p - p'$. Тогда наилучшее пополнение текущего частного решения дает медианных вершин меньше

чем p. Но поскольку очевидно, что передаточное число $\sigma(\bar{X}_p)$ оптимальной p-медианы $\bar{X}_p$ монотонно убывает с увеличением p, то при $h < p - p'$ текущее частное решение наверняка не является частью оптимального p-медианного решения и тогда можно выполнить шаг возвращения.

5.3. Другой алгоритм направленного поиска

Рассмотрение альтернативных распределений, возникающих при выборе конкретных значений для переменных ξ_{ij}, приводит в процедуре поиска из разд. 5.2 к частным подзадачам. Укажем еще одну процедуру поиска [18]; она состоит в построении бинарного дерева поиска, причем это построение осуществляется путем последовательной проверки условия: является или нет данная вершина медианной вершиной? Процедура поиска в этом случае задается множествами S^+, S^- и F. Первое множество соответствует тем вершинам, которые в текущий момент являются медианными, второе соответствует немедианным вершинам, а третье — вершинам, о которых пока нельзя сказать ничего определенного.

Нижняя граница стоимости оптимального решения, определяемая уже осуществленным распределением некоторых вершин по множествам S^+ или S^-, может быть вычислена [18] способом, аналогичным приведенному в предыдущем разделе.

Итак, вершина $x_j \in S^-$ должна быть прикреплена к некоторой вершине множества $S^+ \cup F$. Наилучшее из этих распределений имеет стоимость

$$u'_j = \min_{x_i \in S^+ \cup F} [v_j d(x_j, x_i)] \tag{6.21}$$

С другой стороны, вершина $x_j \in F$, которая не может стать медианной вершиной, должна обладать стоимостью, равной по крайней мере

$$u''_j = \min_{\substack{i \neq j \\ x_i \in S^+ \cup F}} [v_j d(x_j, x_i)]. \tag{6.22}$$

Нижняя граница вычисляется по формуле

$$\sum_{x_j \in S^-} u'_j + U'', \tag{6.23}$$

где U'' есть сумма $n - p - |S^-|$ наименьших чисел u''_j. Следует отметить, что $n - p - |S^-|$ есть число вершин, которые пока еще свободны, но могут быть медианными вершинами окончательного решения и, следовательно, должны быть сопоставлены другим вершинам.

В работе [18] оценка из выражения (6.23) используется при древовидном поиске специального типа (равностоимостного). При

таком поиске ветвление производится в узлах дерева, соответствующих наименьшей нижней границе решения. Методы поиска с приоритетом по глубине, аналогичные тем, которые рассматривались в начале настоящего раздела и в которых используются оценки, подобные приведенной в выражении (6.23),— несколько модифицированные, чтобы можно было решать обобщенные p-медианные задачи, возникающие на практике при размещении складов (пунктов обслуживания), — встречаются также в работах [29, 30, 15, 7, 28].

5.4. Приближенный алгоритм

Тэйтц и Барт [32] предложили эвристический метод для нахождения p-медианы. Метод состоит в следующем: случайным образом выбираются p вершин, они и образуют начальное множество S, аппроксимирующее p-медианное множество $\overline{X}_p$. Затем выясняется, может ли некоторая вершина $x_j \in X - S$ заменить вершину $x_i \in S$ (как медианная вершина), для чего строится новое множество $S' = (S \cup \{x_j\}) - \{x_i\}$ и сравниваются передаточные числа $\sigma(S')$ и $\sigma(S)$. Если $\sigma(S') < \sigma(S)$, то вершина x_i замещается вершиной x_j и из множества S получается множество S', которое лучше аппроксимирует p-медианное множество $\overline{X}_p$. Затем исследуется уже множество S', аналогично тому как исследовалось S, и т. д., пока не будет построено множество $\overline{S}$, такое, что ни одну его вершину нельзя заместить вершиной из $X - \overline{S}$ и получить при этом множество с меньшим передаточным числом, чем $\sigma(\overline{S})$. Множество $\overline{S}$ берется в качестве требуемого приближения к множеству $\overline{X}_p$.

5.4.1. Описание алгоритма

Шаг 1. Выбрать некоторое множество S из p вершин в качестве начального приближения к p-медиане. Назовем все вершины $x_j \notin S$ «неопробованными».

Шаг 2. Взять произвольную «неопробованную» вершину и для каждой вершины $x_i \in S$ вычислить «приращение» Δ_{ij}, соответствующее замене вершины x_i вершиной x_j, т. е. вычислить

$$\Delta_{ij} = \sigma(S) - \sigma((S \cup \{x_j\}) - \{x_i\}). \tag{6.24}$$

Шаг 3. Найти $\Delta_i = \max_{x_i \in S} [\Delta_{ij}]$.

(i) Если $\Delta_{i_0 j} \leqslant 0$, то назвать вершину x_j «опробованной» и перейти к шагу 2.

(ii) Если $\Delta_{i_0 j} > 0$, то $S \leftarrow (S \cup \{x_j\}) - \{x_i\}$ назвать вершину x_j «опробованной» и перейти к шагу 2.

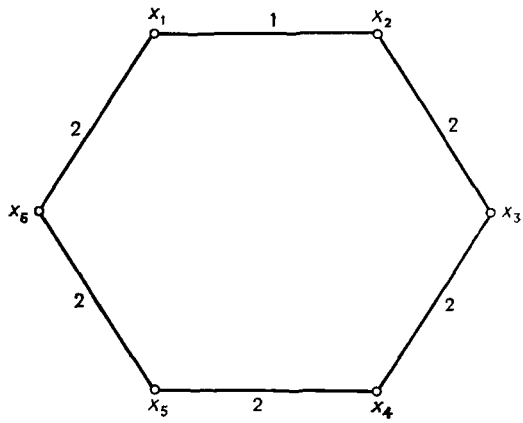


Рис. 6.1. Контрпример к приближенному алгоритму разд. 5.4.

Шаг 4. Повторять шаги 2 и 3 до тех пор, пока все вершины из $X - S$ не будут опробованы. Эта процедура оформляется как цикл. Если при выполнении последнего цикла совсем не будет замещений вершин на шаге 3 (i), то перейти к шагу 5. В противном случае, т. е. если осуществлено некоторое замещение, назвать все вершины «неопробованными» и вернуться к шагу 2.

Шаг 5. Стоп. Текущее множество S является подходящей аппроксимацией p-медианного множества $\overline{X}_p$.

Очевидно, что приведенный выше алгоритм не во всех случаях дает оптимальный ответ [18]. Действительно, рассмотрим неориентированный граф, изображенный на рис. 6.1. Числа, стоящие около ребер, равны соответствующим реберным стоимостям. Считаем, что все вершины имеют единичные веса. Если искать 2-медиану и в качестве начального множества S взять $\{x_3, x_6\}$ с передаточным числом $\sigma(S) = 8$, то никакое замещение только одной вершины не приводит к множеству с меньшим передаточным числом. Однако множество $\{x_3, x_6\}$ не является 2-медианой данного графа. Существуют два 2-медианных множества: $\{x_1, x_4\}$ и $\{x_2, x_5\}$ с передаточными числами $\lambda(\overline{X}_2) = 7$.

Алгоритм, описанный в настоящем разделе, является в действительности лишь одним из целого семейства алгоритмов, базирующихся на локальной оптимизации и на идее λ-оптимальности, впервые введенной Лином [22] для задачи коммивояжера, а впоследствии развитой и использованной в других работах [3, 4, 19] при исследовании различных комбинаторных проблем.

Вернемся к нашей задаче. Множество S из p вершин называется λ-оптимальным ($\lambda \leqslant p$), если замена любых λ вершин из

множества S любыми λ вершинами, не принадлежащими S, не может породить нового множества с передаточным числом, меньшим, чем у S. Очевидно, что если S является p-медианным множеством $\overline{X}_p$ рассматриваемого графа, то S p-оптимально. Совершенно очевидно также, что если S' является λ-оптимальным множеством и S'' — λ-оптимальное множество, то $\sigma(S'') \leqslant \sigma(S')$ при $\lambda'' > \lambda'$.

Согласно приведенным выше определениям решение, полученное с помощью алгоритма из разд. 5.4.1, можно назвать 1-оптимальным. Подобные алгоритмы можно дать и для порождения 2-оптимальных, 3-оптимальных и т. д. решений. Чтобы убедиться в λ-оптимальности множества S, нужно выполнить

$$\binom{p}{\lambda}\binom{n-p}{\lambda}$$

возможных замещений (а, следовательно, и вычислений передаточных чисел σ). Это число довольно быстро растет с ростом λ, а поэтому такие алгоритмы практически нельзя использовать для значений λ, больших 3.

6. Задачи

1 Доказать, что каково бы ни было множество Y_p, состоящее из p точек, лежащих на ребрах или являющихся вершинами графа G, найдется хотя бы одно подмножество $X_p \subset X$, удовлетворяющее условию: $|X_p| = p$ и $\sigma(X_p) \leqslant \sigma(Y_p)$.

2. Для графа, изображенного на рис. 6.2, найти медианы, 2-медианы, 3-медианы и 4-медианы.

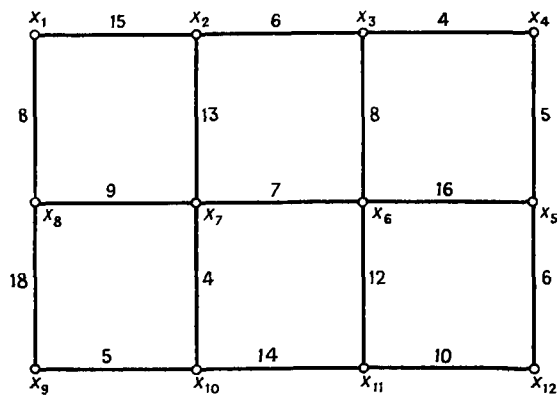


Рис. 6.2.

3. Итерационный процесс для решения задачи построения 1-медианы в случае бесконечного графа, вложенного в евклидову плоскость, состоит в следующем [8].

Пусть G — бесконечный полный граф, представляющий евклидову плоскость, и пусть S — заданное подмножество вершин, состоящее из n точек с координатами (x_i, y_i), $i = 1, 2, \ldots, n$. Требуется выбрать некоторую точку (x_m, y_m) для расположения в ней медианной вершины и чтобы при этом выражение

$$\sigma_m = \sum_{i=1}^{n} v_i d_{im}$$

было минимальным; здесь v_i — «вес» точки с координатами (x_i, y_i) и d_{im} евклидово расстояние:

$$d_{im} = \sqrt{[(x_i - x_m)^2 + (y_i - y_m)^2]}.$$

В точке минимума мы имеем:

$$\frac{\partial \sigma_m}{\partial x_m} = \sum_{i=1}^{n} \left[\frac{v_i (x_i - x_m)}{d_{im}}\right] = 0 \qquad (1)$$

и

$$\frac{\partial \sigma_m}{\partial y_m} = \sum_{i=1}^{n} \left[\frac{v_i (y_i - y_m)}{d_{im}}\right] = 0. \qquad (2)$$

Эти уравнения можно решить итерационным методом, представив их предварительно в следующем виде:

$$x_m = \sum_{i=1}^{n} \left[\frac{v_i x_i}{d_{im}}\right] \Big/ \sum_{i=1}^{n} \left[\frac{v_i}{d_{im}}\right], \qquad (3)$$

$$y_m = \sum_{i=1}^{n} \left[\frac{v_i y_i}{d_{im}}\right] \Big/ \sum_{i=1}^{n} \left[\frac{v_i}{d_{im}}\right]. \qquad (4)$$

Начиная с некоторой точки (x_m, y_m), можно вычислить расстояния d_{im}, а затем, используя выражения (3) и (4), найти новое лучшее приближение для (x_m, y_m). Затем заново вычисляются расстояния d_{im} и координаты x_m и y_m. Процесс продолжается до тех пор, пока не стабилизируются величины x_m и y_m.

Используя описанный итерационный метод, найти 1-медиану в том случае, когда множество S состоит из следующих точек: (2,2), (2,6), (4,8), (5,5), (6,1), (8,6) и (9,3) с весами для всех вершин, равными $v_i = 1$.

Показать, что результат в этом итерационном процессе при нахождении глобального оптимума для (x_m, y_m) не зависит от выбора начального расположения этой точки.

4. Распространить приведенную выше итерационную схему на общий случай нахождения p-медианы. Позволяет ли этот обобщенный метод получать глобальный оптимум для p-медианы?

5. Пусть граф $G = (X, \Gamma)$ является деревом и после удаления ребра (x_i, x_j) распадается на две связные компоненты $G_i = (X_i, \Gamma)$ и $G_j = (X_j, \Gamma)$, где $x_i \in X_i$, $x_j \in X_j$ и $X_i \cup X_j = X$. Введем обозначение:

$$v(S) = \sum_{x_i \in S} v_i.$$

Показать, что медиана $\bar{x}$ графа G удовлетворяет условиям:

(i) если $v(X_i) \geqslant v(X_j)$, то $\bar{x} \in X_i$.

(ii) если $v(X_i) \leqslant v(X_j)$, то $\bar{x}$ не меняется при замене графа G графом G_j, у которого вес вершины x_j полагается равным $v_j + v(X_i)$. Учитывая эти факты, дать алгоритм для нахождения медианной вершины в дереве, который был бы более эффективным, чем непосредственное использование уравнений (6.2) и (6.3) (см. [13]).

7. Список литературы

1. Beale E. M. L. (1970), Selecting an optimum subset, Integer and nonlinear programming, Abadie, Ed., North Holland, Amsterdam.
2. Cabot A. V., Francis R. L., Stary M. A. (1970), A network flow solution to a rectilinear distance facility location problem, *AIIE Trans.*, 2, p. 132.
3. Christofides N., Eilon S. (1969), An algorithm for the vehicle dispatching problem, *Opl. Res. Quart.*, 20, p. 309.
4. Christofides N., Eilon S. (1972), Algorithms for large scale travelling salesman problems, *Opl. Res. Quart.*, 23, p. 511.
5. Cooper L. (1963), Location-allocation problems, *Ops. Res.*, 11, p. 331.
6. Curry G. R., Skeith R. W. (1969), A dynamic programming algorithm for facility location and allocation, *AIIE Trans.*, 1, p. 133.
7. Efroymson E., Ray T. L. (1966), A branch and bound algorithm for plant location, *Ops. Res.*, 14, p. 361.
8. Eilon S., Watson-Gandy C. D. T., Christofides N. (1971), Distribution management: mathematical modelling and practical analysis, Griffin, London.
9. Ellwein L. B., Gray P. (1971), Solving fixed charge location-allocation problems with capacity and configuration constraints, *AIIE Trans.*, 3, p. 290.
10. Elson D. G. (1972), Site location via mixed-integer programming, *Opl. Res. Quart.*, 23, p. 31.
11. Garfinkel R. S., Nemhauser G. L. (1972), Integer programming, Wiley, New York.
12. Goldman A. J. (1969), Optimal locations for centres in a network, *Trans. Sci.*, 3, p. 352.
13. Goldman A. J. (1970), Optimal centre location in simple networks, Paper FP 6.1, presented at the 38th National Meeting of the Operations Research Soc. of America, Detroit.
14. Goldman A. J., Mayers P. R. (1965), A domination theorem for optimal locations, *Ops. Res.*, 13, 13—147 (Abstract).

15. Gray P. (1967), Mixed integer programming algorithm for site selection and other fixed charge problems having capacity constraints, Department of Operations Research, Report No. 6, Stanford University.
16. Hakimi S. L. (1965), Optimum distribution of switching centres in a communications network and some related graph theoretical problems.
17. Hakimi S. L. (1964), Optimum locations of switching centres and the absolute centres and medians of a graph, *Ops. Res.*, **12**, p. 450.
18. Jarvinen P., Rajala J., Sinervo H. (1972), A branch and bound algorithm for seeking the *p*-median, *Ops. Res.*, **20**, p. 173.
19. Kerningan B. W., Lin S. (1970), An efficient heuristic procedure for partioning graphs, *Bell Syst. Tech. J.*, **49**, p. 291.
20. Levy J. (1967), An extended theorem for location on a network, *Opl. Res. Quart.*, **18**, p. 433.
21. Lin S. (1965), Computer solutions of the travelling salesman problem, *Bell. Syst. Tech. J.*, **44**, p. 2245.
22. Maranzana F. E. (1964), On the location of supply points to minimize transport costs, *Opl. Res. Quart.*, **15**, p. 261.
23. Marsten R. E. (1972), An algorithm for finding almost all the *p*-medians of a network, Paper 23, Northwestern University, Evaston, Illinois.
24. Rao M. R. (1972), The rectilinear facilities location problem, Working paper No. F7215, The graduate school of management, University of Rochester, Rochester, New York.
25. Revelle C. S., Marks D., Liebman J. C. (1970), An analysis of private and public sector location models, *Man. Sci.*, **16**, p. 692.
26. Revelle C. S., Swain R. W. (1970), Central facilities location, *Geographical Analysis*, **2**, p. 30.
27. Rojeski P., Revelle C. S. (1970), Central facilities location under an investment constraint, *Geographical Analysis*, **2**.
28. Sá G. (1964), Branch and bound and approximate solutions to the capacitated plant-location problem, *Ops. Res.*, **17**, p. 1005.
29. Spielberg K. (1969), An algorithm for the simple plant location problem with some side conditions, *Ops. Res.*, **17**, p. 85.
30. Spielberg K. (1969), Plant location with generalized search origin, *Man. Sci.*, **16**, p. 165.
31. Surkis J. (1967), Optimal warehouse location, XIV Int. TIMS Conf., Mexico City, Mexico.
32. Teitz M. B., Bart P. (1968), Heuristic methods for estimating the generalized vertex median of a weighted graph, *Ops. Res.*, **16**, p. 955.

Глава 7

ДЕРЕВЬЯ

1. Введение

Одним из наиболее важных понятий теории графов, которое к тому же часто встречается в областях, не имеющих на первый взгляд никакого отношения к графам, является дерево. В настоящей главе вводятся понятия ориентированного и неориентированного деревьев и дается краткое описание тех областей, где они применяются. Некоторые из этих приложений более подробно рассматриваются в других главах книги.

Определения. Нижеследующие определения неориентированного дерева, как легко показать, эквивалентны друг другу. (См. эадачу 7.1.)

Неориентированное дерево есть:

(i) связный граф, содержащий n вершин и $n-1$ ребер, либо

(ii) связный граф, не имеющий циклов, либо

(iii) граф, в котором каждая пара вершин соединена одной и только одной простой цепью.

Если $G = (X, A)$ — неориентированный граф с n вершинами, то *остовным деревом* (или, короче, *остовом*) *графа* G называется всякий остовный подграф графа G, являющийся деревом (в смысле приведенного выше определения). Например, если G — граф, показанный на рис. 7.1а, то граф на рис. 7.1б является остовом графа G, как и граф, изображенный на рис. 7.1в. Из сформулированных выше определений вытекает, что остов графа G можно также рассматривать как минимальный связный остовный подграф графа G, где «минимальность» понимается в том смысле, как говорилось в гл. 2, т. е. никакое собственное подмножество ребер этого остова не образует связный остовный подграф графа G.

Понятие дерева как математического объекта было впервые предложено Кирхгофом [36] в связи с определением фундаментальных циклов, применяемых при анализе электрических цепей. Приблизительно десятью годами позже Кэли [5] вновь (независимо от Кирхгофа) ввел понятие дерева и получил бо́льшую часть первых результатов в области исследования свойств деревьев.

Ориентированное дерево (называемое также *древовидностью*) определяется аналогичным образом.

Определение. Ориентированное дерево представляет собой ориентированный граф без циклов, в котором полустепень захода каждой вершины, за исключением одной (например, вершины x_1), равна единице, а полустепень захода вершины x_1 (называемой корнем этого дерева) равна нулю [1].

На рис. 7.2 показан граф, который является ориентированным деревом с корнем в вершине x_1. Из приведенного определения

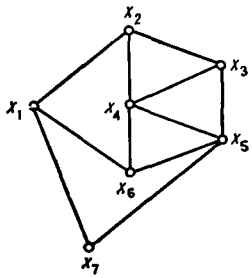


Рис. 7.1а. Граф G.

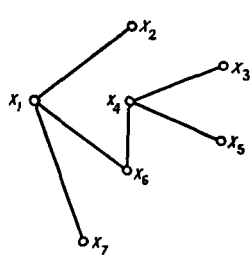


Рис. 7.1б. Остов графа G.

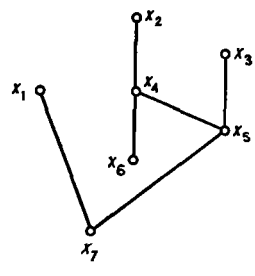


Рис. 7.1в. Другой остов графа G.

следует, что ориентированное дерево с n вершинами имеет $n - 1$ дуг и связно. Также очевидно, что не всякий ориентированный граф содержит остовное ориентированное дерево. Это подтверждает граф, изображенный на рис. 7.3.

Следует отметить, что неориентированное дерево можно преобразовать в ориентированное: надо взять его произвольную вершину в качестве корня и ребрам приписать такую ориентацию,

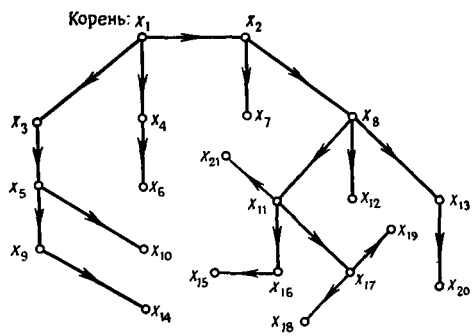


Рис. 7.2. Ориентированное дерево.

[1]) Такое дерево часто называют *выходящим* (или *корневым*).— *Прим. ред.*

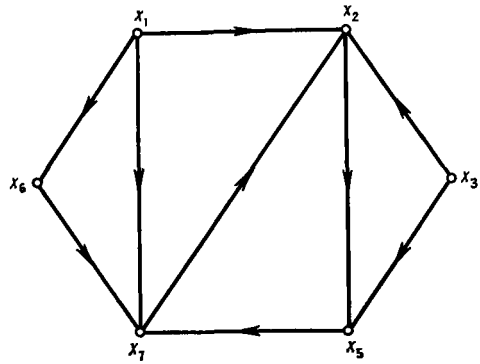


Рис. 7.3. Граф без ориентированного остовного графа.

чтобы каждая вершина соединялась с корнем (только одной) простой цепью. Обратно, если $T = (X, B)$ — ориентированное дерево, то $\overline{T} = (X, \overline{B})$, где $\overline{B}$ — множество дуг дерева T без учета их ориентации, является неориентированным деревом.

«Генеалогическое дерево», в котором вершины соответствуют лицам мужского пола, а дуги ориентированы от родителей к детям, представляет собой хорошо известный пример ориентированного дерева. Корень в этом дереве соответствует «основателю» рода (лицу, родившемуся раньше остальных).

В настоящей главе приводится алгоритм порождения всех остовных деревьев произвольного неориентированного графа и даются методы прямого построения кратчайших остовных деревьев во взвешенном графе (в котором веса приписаны дугам). Кратчайшее остовное дерево (SST) графа находит, очевидно, применение при прокладке дорог (газопроводов, линий электропередач и т. д.), когда необходимо связать n точек некоторой сетью так, чтобы общая длина «линий связи» была минимальной. Если точки лежат на евклидовой плоскости, то их можно считать вершинами полного графа G с весами дуг, равными соответствующим «прямолинейным» расстояниям между концевыми точками дуг. Тогда, поскольку «разветвление» дорог допускается только в заданных n точках, SST графа G будет как раз требуемой сетью дорог, имеющей наименьший вес.

Если же «разветвление» дорог можно производить и «вне» заданных n точек, то возможна и более «короткая» (с меньшей стоимостью) сеть. Нахождение такой сети дорог представляет собой хорошо известную задачу Штейнера. Краткому обсуждению этой задачи посвящен последний раздел данной главы.

2. Построение всех остовных деревьев графа

В некоторых ситуациях возникает необходимость в построении полного списка остовных деревьев графа G. Например, в том случае, когда надо отобрать «наилучшее» дерево, а критерий, позволяющий осуществить такой отбор, является очень сложным (или даже частично субъективным), так что непосредственное решение задачи оптимизации (не использующее перечисление всех остовных деревьев) оказывается невыполнимым. В других ситуациях, например при нахождении передаточной функции системы [42,6] или при вычислении определителей некоторых матриц в макроэкономической теории [2], с помощью порождения всех остовов соответствующих графов можно добиться упрощения вычислительных процедур.

Число различных остовов полного связного неориентированного помеченного графа с n вершинами было найдено впервые Кэли [4]. Оно равно n^{n-2}. У Муна [45] приводится список из более чем 25 работ, содержащих разнообразные доказательства этой формулы. (См. также задачу 6.) Формулы для числа остовов в более общих графах можно найти у Риордана [49]. Хотя эти формулы, как правило, очень сложные и их вывод для наших целей не нужен, но стоит, пожалуй, привести следующий результат.

Теорема 1. *Пусть G — n-вершинный граф без петель и B_0 — его матрица инциденций с одной удаленной строкой (т. е. с $n-1$ независимыми строками). Пусть B_0^t — транспонированная матрица к B_0. Тогда определитель $|B_0 \cdot B_0^t|$ равен числу различных остовных деревьев графа G.*

Доказательство этой теоремы можно найти в [53] и в [1] (см. также задачу 5).

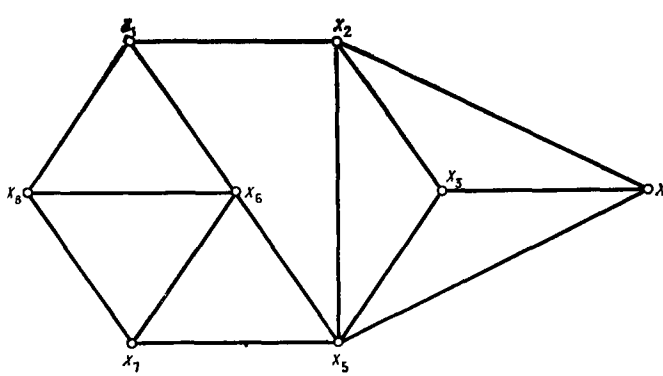


Рис. 7.4. Граф G.

2.1. Элементарные преобразования деревьев

Рассмотрим два (ориентированных или неориентированных) остовных дерева $T_1 = (X, A_1)$ и $T_2 = (X, A_2)$ графа $G = (X, A)$. «Расстояние» между двумя деревьями обозначается через $d(T_1, T_2)$ и определяется как число дуг из T_1, которых нет в T_2 (или, что эквивалентно, как число дуг из T_2, которых нет в T_1, поскольку оба дерева T_1 и T_2 имеют $(n-1)$ дуг). Если $d(T_1, T_2) = 1$, т. е. если

$$(A_1 \cup A_2) - (A_1 \cap A_2) = \{a_1, a_2\},$$

где $a_1 \in A_1$ и $a_2 \in A_2$, то дерево T_2 можно получить из дерева T_1, удалив из T_1 дугу a_1 и добавив дугу a_2. Такое преобразование дерева T_1 в дерево T_2 называется *элементарным преобразованием дерева.*

Теорема 2. *Если T_0 и T_k — остовные деревья графа и $d(T_0, T_k) = k$, то дерево T_k может быть получено из T_0 с помощью серий из k элементарных преобразований.*

Доказательство. Пусть $a_0^1, a_0^2, \ldots, a_0^k$ — k дуг из T_0, которых нет в T_k, и $a_k^1, a_k^2, \ldots a_k^k$ — k дуг из T_k, которых нет в T_0. Если в дерево T_0 добавить дугу a_k^1, то в получившемся графе, согласно определению дерева, найдется цикл. (На рис. 7.5(а) жирными линиями показано неориентированное дерево T_0, а пунктирной линией — дуга $a_k^1 = (x_5, x_6)$ дерева T_4, приведенного на рис. 7.5(д).) В полученном цикле содержится по крайней мере одна дуга, не принадлежащая дереву T_k. Следовательно, ее можно удалить, разорвав тем самым цикл, и это даст новое дерево T_1. Поскольку в T_1 число дуг, общих с дугами из T_k, на единицу больше (чем в T_0), то $d(T_1, T_k) = k - 1$. Применяя элементарные преобразования дальше, получим последовательность деревьев $T_2, T_3, \ldots, T_{k-1}, T_k$, в которой $d(T_i, T_{i+1}) = 1$ для каждого $i = 2, \ldots, k-1$. На рис. 7.5(б)—(д) показано, как с помощью 4 элементарных преобразований из дерева T_0 получается дерево T_4.

2.2. Процедура порождения всех деревьев графа

Поскольку, как уже упоминалось в начале этого раздела, число остовных деревьев графа очень быстро растет с ростом числа его дуг, то, очевидно, нужен эффективный метод порождения исчерпывающего, но без повторений, списка остовных деревьев. Один из таких способов состоит в использовании элементарных преобразований деревьев, описанных в предыдущем разделе, для последовательного порождения остовов, начиная с некоторого начального остова T_0. Методы, основанные на этом принципе, даны в работах

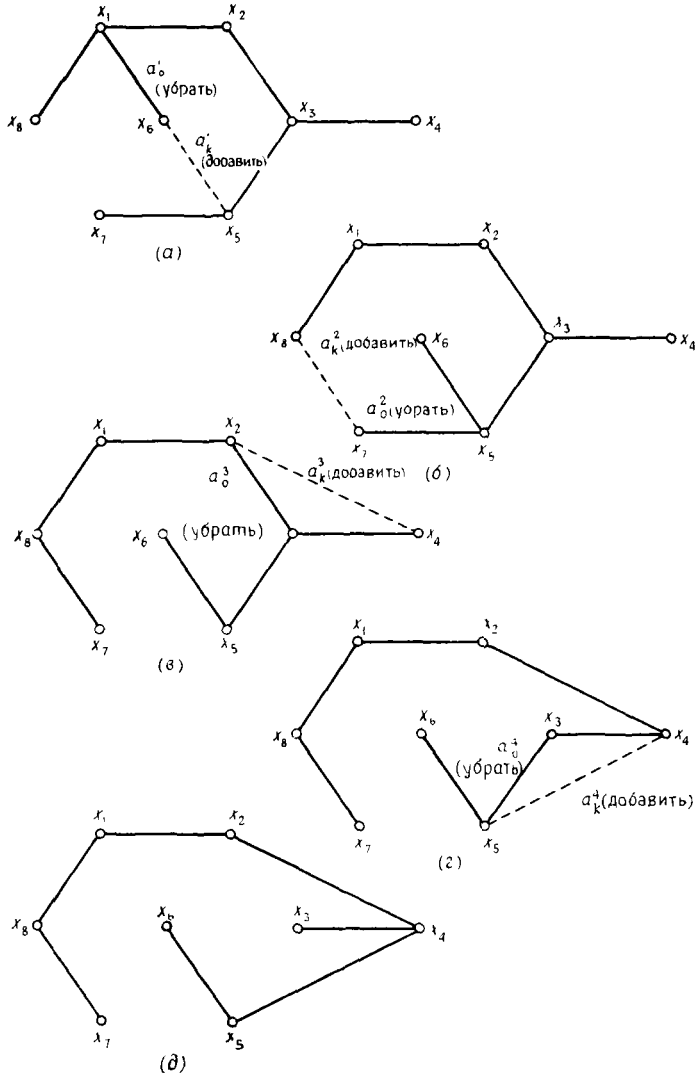


Рис. 7.5. Порождение T_4 из T_0 посредством элементарных преобразований деревьев. (а) Остов T_0 графа с рис. 7.4. (б) Остов T_1. (в) Остов T_2. (г) Остов T_3. (д) Остов T_k ($k = 4$).

Маеды [42], Пауля [47], Чена [8] и других [20]. Однако процедурам, опирающимся на элементарные преобразования деревьев, присущ следующий недостаток: для порождения нового дерева необходимо привлекать все найденные ранее деревья. Число деревьев становится столь большим, что для хранения их необходимо (с вычислительной точки зрения) использовать вспомогательные устройства памяти, так как быстродействующая оперативная память для этой цели мала. Поскольку при обращении к вспомогательным устройствам памяти значительно возрастает время вычисления, то любой метод, базирующийся на элементарных преобразованиях деревьев, оказывается неэффективным, если не будут применяться такие преобразования, в которых используется только последнее из полученных остовных деревьев. (См. раздел 2.4.) Очевидно, что наилучшим будет такой алгоритм порождения всех остовов графа, когда список остовов строится без повторений и построенные остовы записываются во вспомогательную память, но в процессе работы алгоритма из этой памяти ничего не берется.

В следующем разделе будет описан один такой алгоритм, основанный на поиске, использующем дерево решений. В работах [10, 9, 43] даются другие алгоритмы, базирующиеся на порождении всех главных квадратных подматриц приведенной матрицы инциденций B_0, о которой упоминалось в теореме 1.

2.2.1. Основной метод. Мы приводим описание алгоритма для случая неориентированного графа, но его распространение на ориентированные графы — для порождения всех остовных древовидностей ориентированного графа — совершенно очевидно.

Первый шаг метода состоит в приписывании номеров ребрам графа G (ребра нумеруются от 1 до m, где m — число ребер в графе G). На каждом этапе (т. е. при каждом ветвлении в дереве решений) выбирается ребро, которое вместе с остальными, выбранными уже на предыдущих этапах, будет образовывать часть конструируемого дерева. Таким образом, прежде чем отобрать такое ребро, выясняют, действительно ли добавление его к частично сформированному дереву (которое на этом шаге является набором поддеревьев) не приводит к появлению цикла. Если цикл появляется, то данное ребро отбрасывается и проверке подвергается следующее ребро, с большим номером. Если цикла нет, то ребро добавляется к другим, уже отобранным, и процесс продолжается до тех пор, пока не будет построено остовное дерево. Ребра перебираются в порядке возрастания их номеров; это приводит к исчерпывающему и без повторений решению задачи.

Для облегчения манипуляций с поддеревьями в каждом поддереве выделяют произвольным образом корень (некоторую вершину поддерева) и затем рассматривают поддерево уже как древо-

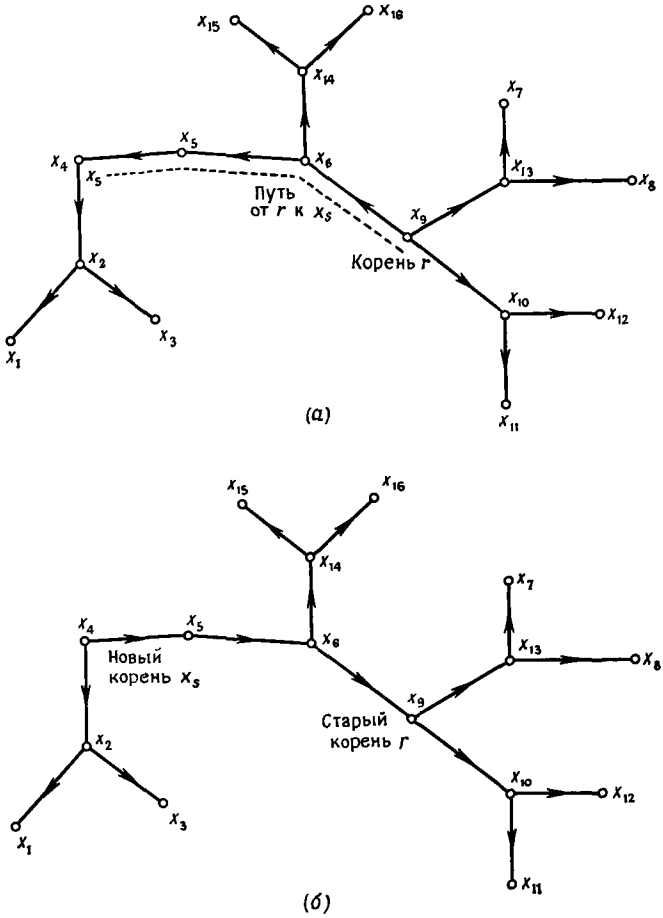


Рис. 7.6. Замена корня x_9 из (а) на x_4 из (б).

видность. Для организации проверки на возможность образования цикла (при добавлении ребра) каждую вершину x_j помечают парой (r_j, p_j). Процедуры выявления циклов, использующие пометки вершин, встречаются у Джонсона [32], Шринивасана и Томпсона [52], Гловера и Клингмана [25]. Первая пометка r_j указывает «корень» поддерева, содержащего вершину x_j. Первоначально $r_j = x_j$ для всех вершин x_j. На некотором шаге два поддерева T_1 и T_2 сращиваются посредством добавления ребра $a_l = (x_\alpha, x_\beta)$ с вершиной x_α из T_1 и вершиной x_β из T_2. Если на этом шаге r_1 — «корневая» пометка вершин в T_1, а r_2 — «корневая» пометка вершин в T_2 и $r_1 < r_2$ (например), то все вершины в T_2 должны «сме-

нить» свои корневые пометки на r_1 и два поддерева T_1 и T_2 «сольются» в единственное новое дерево T_1.

Вторая пометка p_j, приписанная вершине x_j, указывает вершину, предшествующую вершине x_j, т. е. если (x_k, x_j) — дуга рассматриваемого поддерева, то $p_j = x_k$. Для корневой вершины дерева такая пометка полагается равной нулю. Для дерева, изображенного на рис. 7.6 (а), $p_5 = x_6$, $p_{13} = x_9$ и $p_9 = 0$.

А. *Замена корня дерева*

Если корнем дерева T является вершина r и нужно в качестве корня выбрать новую вершину x_s, то такую «замену» r на x_s можно осуществить простым обращением ориентации дуг, принадлежащих цепи, идущей от r к x_s, не меняя при этом ориентацию других дуг (см. рис. 7.6). Соответствующие изменения пометок будут таковы.

Изменение пометок «предшествования»

1. Пусть $x_j = x_s$ и $z = p_j$.
2. Положить $x_i = z$, а $z = p_i$.
3. Шаг обновления: $p_i = x_j$.
4. Если $x_i = r$, то перейти к шагу 5, в противном случае положить $x_j = x_i$ и перейти к шагу 2.
5. Положить $p_s = 0$, стоп.

Изменение корневых пометок

1. У всех вершин, имеющих корневую пометку r, заменить ее на пометку x_s.

Б. *Сращивание двух поддеревьев*

Если осуществлено сращивание двух поддеревьев T_1 и T_2 (путем добавления ребра (x_α, x_β), как было описано ранее), то в пометки необходимо внести следующие изменения:

(i) У вершин с корневой пометкой r_2 заменить эту пометку на r_1.

(ii) Заменить в дереве T_2 корень r_2 на x_β (так, как было описано выше в пункте А), после чего изменить пометку «предшествования» у вершины x_β с $p_\beta = 0$ на $p_\beta = x_\alpha$.

В. *Расщепление дерева на две части*

Поскольку метод порождения деревьев, рассматриваемый ниже, является поиском, использующим дерево решений, то возникает необходимость удаления некоторых ребер (на шагах возвращения), чтобы испытать затем другие ребра. В такой ситуации удаление ребра приводит к расщеплению некоторого дерева на две части, например на T_1 и T_2, и в пометки одного из этих поддеревьев должны быть внесены изменения. Пусть удаляется ребро (x_α, x_β), где $x_\alpha \in T_1$ и $x_\beta \in T_2$. Тогда при $p_\beta = x_\alpha$ (т. е. если ребро

(x_α, x_β) в первоначальном дереве ориентировано от x_α к x_β) пометки в дереве T_1 можно оставить прежними, а пометки в дереве T_2 должны быть изменены. Если же $p_\alpha = x_\beta$ (т. е. ребро (x_α, x_β) ориентировано от x_β к x_α), то можно не менять пометки в дереве T_2, но нужно изменить пометки в T_1. Предполагая, что пометки меняются в дереве T_2, покажем, как надо «восстанавливать» корень в этом дереве.

1. Положить $S = \{x_\beta\}$ и $p_\beta = 0$ (x_β будет корнем дерева T_2).
2. Найти все вершины x_j с $p_j \in S$ и изменить их корневые пометки на $r_j = x_\beta$. Если таких вершин нет, остановиться.
3. Шаг обновления: $S = S \cup \{x_j \mid p_j \in S\}$; вернуться к шагу 2.

Следует отметить, что ни у одной вершины, кроме нового корня x_β, пометки предшествования менять не нужно. Заметим также, что число описанных выше шагов 2 и 3, которое необходимо для восстановления корня, равно длине самой длинной цепи в T_2, исходящей из вершины x_β.

2.2.2. Описание алгоритма. Возьмем произвольную вершину x^* графа G. Пусть ее степень равна d^*. Перенумеруем ребра, инцидентные этой вершине: $a_1, a_2, \ldots, a_{d^*}$. Затем перенумеруем остальные ребра графа G: $a_{d^*+1}, \ldots, a_m$. При порождении деревьев ребра будут перебираться в соответствии с введенной нумерацией.

Шаг 1. Приписать вершинам пометки: (r_i, p_i), где $r_i = x_i$ и $p_i = 0$, $\forall x_i \in X$. Положить $k = 1$.

Шаг 2. Выбрать для исследования некоторое ребро. Например, $a_k = (x_i, x_j)$. Если $k \leqslant m$, где m — число ребер графа, то перейти к 2 (i). При $k = m + 1$, т. е. если «неисследованных» ребер нет, перейти к шагу 5.

(i) Если $r_i = r_j$, то это означает, что вершины x_i и x_j принадлежат одному и тому же поддереву и добавление ребра a_k приведет к появлению цикла. Отбросить ребро a_k, т. е. положить $k = k + 1$ и вернуться к шагу 2.

(ii) Если $r_i \neq r_j$, то ребро a_k можно добавить к ребрам построенных поддеревьев. Перейти к шагу 3.

Шаг 3. Срастить два поддерева, у которых вершины имеют корневые пометки r_i и r_j, применив для этого метод, описанный выше в пункте Б.

Шаг 4. Отобрав $n - 1$ ребер, мы получаем некоторое дерево. Запомнить это дерево и перейти к шагу 5. Если отобрано меньше, чем $n - 1$ ребер, то положить $k = k + 1$ и вернуться к шагу 2.

Шаг 5. (Возвращение.) Удалить ребро, добавленное последним. Предположим, что таким ребром является a_l. Если a_l —

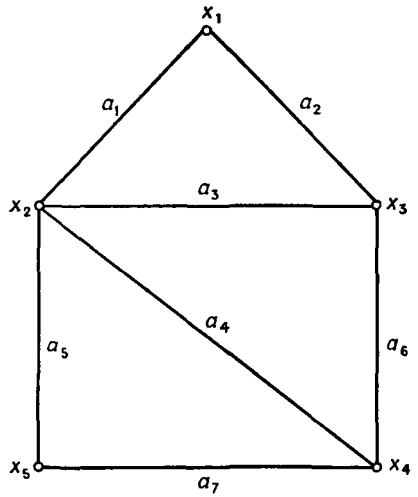


Рис. 7.7. Граф из примера 2.3.

единственное оставшееся для добавления ребро, $l = d^*$, то остановиться. Все остовные деревья, таким образом, построены. (При любом дальнейшем ветвлении дерева решений вершина x^* останется изолированной.)

В противном случае надо обновить пометки, действуя так, как указано в пункте В, положить $k = l + 1$ и возвратиться к шагу 2.

2.3. Пример

Нам нужно построить все остовные деревья графа, изображенного на рис. 7.7. Выберем в качестве x^* вершину x_1; имеем $d^* = 2$. Обозначим ребро (x_1, x_2) через a_1, а ребро (x_1, x_3) через a_2. Остальные ребра перенумеруем, например так, как показано на рис. 7.7 (ребра a_3, a_4, . . ., a_7).

На рис. 7.8 изображено соответствующее дерево решений (оно порождено в процессе работы алгоритма, приведенного в разд. 2.2.2). Если взять ребра, указанные в кружочках какой-либо цепи, выходящей из верхнего узла этого дерева и оканчивающейся в самом нижнем узле, то из них можно построить некоторый остов данного графа. Эти остовы перенумерованы числами от 1 до 21 и приведены на рис. 7.9.

Легко проверить, что у графа, изображенного на рис. 7.7, действительно 21 остов. Это можно установить с помощью теоремы 1 из разд. 2. Матрица инциденций B данного графа имеет следующий вид (считаем, что каждое ребро ориентировано от его конце-

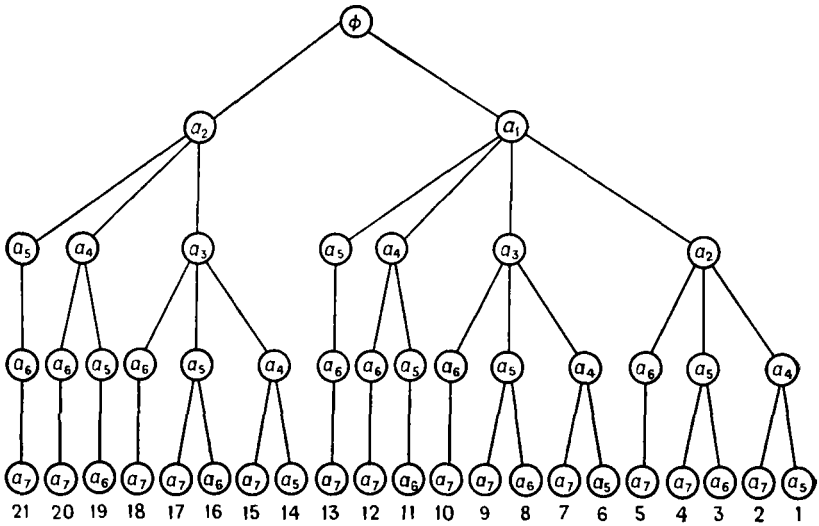


Рис. 7.8. Полное дерево поиска из примера 2.3.

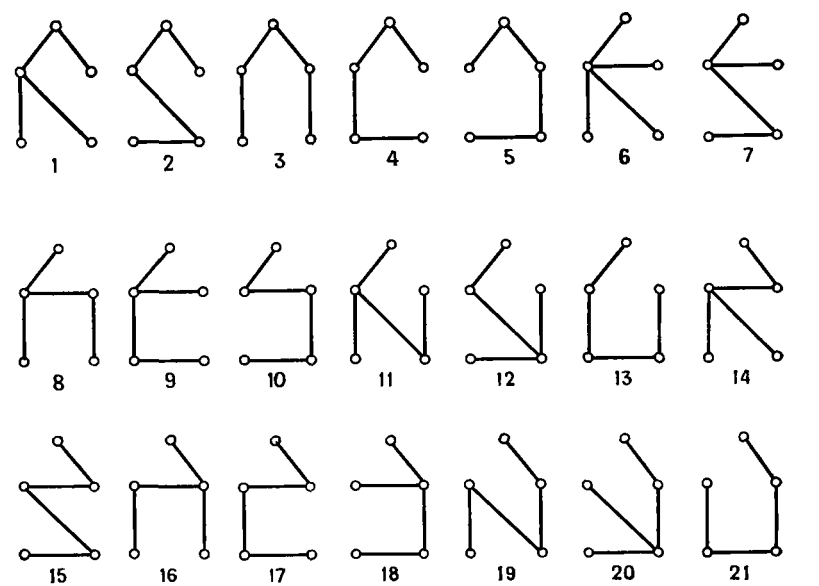


Рис. 7.9. Все остовы графа с рис. 7.7.

вой вершины с меньшим индексом к вершине с большим индексом):

$$B = \begin{array}{c|ccccccc} & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 \\ \hline x_1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ x_2 & -1 & 0 & 1 & 1 & 1 & 0 & 0 \\ x_3 & 0 & -1 & -1 & 0 & 0 & 1 & 0 \\ x_4 & 0 & 0 & 0 & -1 & 0 & -1 & 1 \\ x_5 & 0 & 0 & 0 & 0 & -1 & 0 & -1 \end{array}$$

Удаляя, например, строку x_2, получим матрицу B_o. Произведение матриц $B_o \cdot B_o^t$ выглядит так:

$$B_o \cdot B_o^t = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 2 & -1 & 0 & 0 \\ 2 & -1 & 3 & -1 & 0 \\ 3 & 0 & -1 & 3 & -1 \\ 4 & 0 & 0 & -1 & 2 \end{array}$$

Определитель $|B_o \cdot B_o^t|$ равен 21. Следовательно, по теореме 1 число остовных деревьев данного графа равно 21.

2.4. Граф остовов

Каждому остову графа G сопоставим определенную вершину (нового графа). Две вершины в новом графе соединяются ребром только тогда, когда соответствующие им остовы графа G являются соседними (т. е. расстояние между этими остовами, определяемое в соответствии с разд. 2.1, равно единице). Граф, построенный таким образом, называется *графом остовов* (графа G). Для графа G, изображенного на рис. 7.10 (а), полный список остовов приведен на рис. 7.10 (б), а граф остовов — на рис. 7.10 (в).

Каммингс [15] и Шэнк [51] доказали, что граф остовов любого связного графа является гамильтоновым. Позже Киси и Кадзитани [37, 38] и Камаэ [33] разработали методы нахождения гамильтоновых циклов в графе остовов (и, следовательно, методы построения всех остовов графа G). Эти методы представляют в основном теоретический интерес и не эффективны с вычислительной точки зрения.

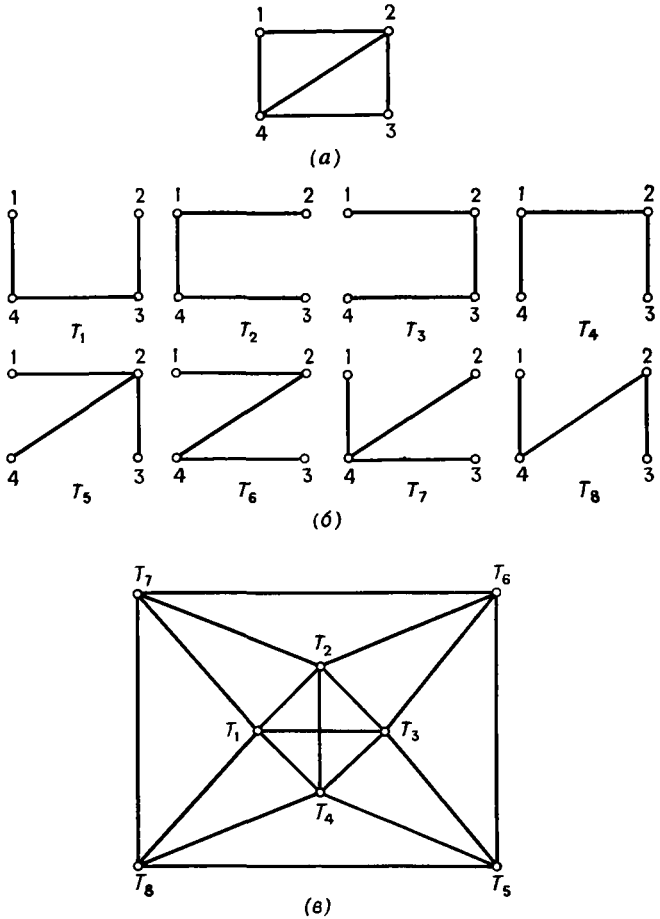
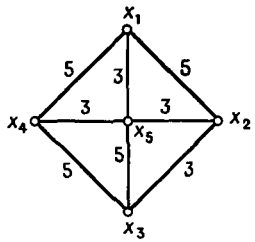


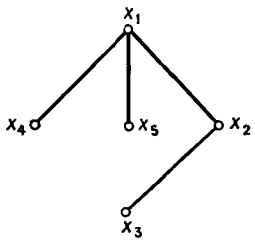
Рис. 7.10. Граф G и его граф остовов.
(а) Граф G. (б) Остовы графа G. (в) Граф остовов графа G.

3. Кратчайший остов (SST) графа

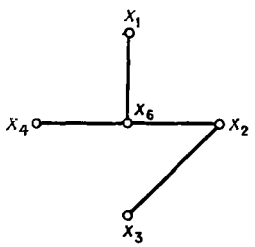
Рассмотрим взвешенный связный неориентированный граф $G = (X, A)$; вес ребра (x_i, x_j) обозначим c_{ij}. Из большого числа остовов графа нужно найти один, у которого сумма весов ребер наименьшая. Такая задача возникает, например, в том случае, когда вершины являются клеммами электрической сети, которые должны быть соединены друг с другом с помощью проводов наименьшей общей длины (для уменьшения уровня наводок). Другой пример: вершины представляют города, которые нужно связать



(а)



(б)



(в)

Рис. 7.11.
(а) Граф G.
(б) Дерево кратчайших путей из x_1.
(в) Кратчайший остов графа.

сетью трубопроводов; тогда наименьшая общая длина труб, которая должна быть использована для строительства (при условии, что вне городов «разветвления» трубопроводов не допускаются), определяется кратчайшим остовом соответствующего графа. Не совсем непосредственно, а как промежуточный шаг, кратчайший остов используется при решении задачи о коммивояжере, которая довольно часто встречается на практике и детально рассматривается в гл. 10.

Следует отметить, что SST графа не имеет никакого отношения к дереву, дающему все кратчайшие пути, выходящие из некоторой выбранной вершины. Так для графа, показанного на рис. 7.11(а), где числа, стоящие около ребер, являются их весами, дерево, дающее все кратчайшие пути, выходящие из вершины x_1, изображено на рис. 7.11 (б), а SST — на рис. 7.11 (в).

Задача построения кратчайшего остова (SST) графа является одной из немногих задач теории графов, которые можно считать

полностью решенными. Итак, пусть T_i и T_j — два произвольных поддерева, полученных путем добавления ребер при построении SST. Если символ T_i использовать также для обозначения множества вершин данного поддерева, то Δ_{ij} может быть определено как кратчайшее из расстояний между вершинами из T_i и вершинами из T_j, т. е.

$$\Delta_{ij} = \min_{x_i \in T_i} [\min_{x_j \in T_j} \{c(x_i, x_j)\}], \quad i \neq j. \tag{7.1}$$

Легко показать, что многократное применение нижеследующей операции приводит к построению SST графа.

Операция J: Для поддерева T_s найти такое поддерево T_{j^*}, чтобы $\Delta_{sj^*} = \min_{T_j} [\Delta_{sj}]$. Пусть $(\overline{x_s, x_{j^*}})$ будет тем ребром, вес которого соответствует величине Δ_{sj^*} в выражении (7.1). Тогда ребро $(\overline{x_s, x_{j^*}})$ принадлежит SST и может быть добавлено к другим ребрам частично сформированного SST.

Доказательство. Предположим, что на некотором этапе, например на k-м, ребра в построенных поддеревьях принадлежат окончательному SST, а ребро $(\overline{x_s, x_{j^*}})$, выбранное в соответствии с приведенным выше условием, в SST не содержится. Поскольку поддерево T_s должно быть связано в конце концов согласно определению с некоторым другим поддеревом, то в SST должно существовать ребро (x_i, x_j), такое, что $x_i \in T_s$ и $x_j \notin T_s$. Удалив ребро (x_i, x_j) из SST, мы расщепим это дерево на две связные компоненты, а добавив ребро $(\overline{x_s, x_{j^*}})$, получим новое дерево, более короткое, чем SST, что противоречит определению SST. Таким образом, ребро $(\overline{x_s, x_{j^*}})$ можно добавить к частично сформированному SST на k-м этапе. Затем надо перейти к следующему этапу построения дерева. Нужно отметить, что результат не зависит от выбора поддерева T_s. Поскольку на начальном этапе (т. е. пока еще никакие ребра не выбраны) предположение о принадлежности ребер к SST автоматически выполнено, то многократное применение операции J даст в конце концов SST графа.

Многие методы, позволяющие строить SST графов, основываются на частных случаях описанной выше операции [40, 48, 41, 46, 17, 35, 26]. Первый из таких методов был предложен Краскалом [40].

3.1. Алгоритм Краскала

Шаг 1. Начать с вполне несвязного графа T, содержащего n вершин.

Шаг 2. Упорядочить ребра графа G в порядке неубывания их весов.

Шаг 3. Начав с первого ребра в этом списке, добавлять ребра в графе T, соблюдая условие: такое добавление не должно приводить к появлению цикла в T.

Шаг 4. Повторять шаг 3 до тех пор, пока число ребер в T не станет равным $n-1$. Получившееся дерево является SST графа G.

В этом алгоритме для добавления к частично сформированному дереву выбирается абсолютно кратчайшее допустимое ребро, а не просто кратчайшее ребро между одним поддеревом T, например T_s, и другим каким-либо поддеревом (как это предполагалось в операции J). Так как выбранное ребро является, очевидно, кратчайшим между *некоторым* поддеревом и каким-то другим поддеревом, то правило выбора в этом алгоритме представляет собой частный случай операции J. Однако при выполнении этого алгоритма может возникнуть такая ситуация, когда очередное кратчайшее ребро, выбранное из списка, построенного на шаге 2, будет соединять две вершины одного и того же поддерева. Добавлять это ребро к T нельзя, поскольку такое добавление приводит к появлению цикла в T. Поэтому на шаге 3, прежде чем добавлять ребро к графу T, надо проверить, является ли оно допустимым в указанном выше смысле. Такую проверку можно выполнить более эффективно (путем осуществления одного сравнения с использованием процедуры расстановки пометок, описанной в разд. 2.2.1), абсолютно так же, как это делается на втором шаге алгоритма из разд. 2.2.2.

Больше всего времени необходимо для выполнения шага 2 рассматриваемого алгоритма. Для графа с m ребрами нужно выполнить порядка $m \log_2 m$ операций, чтобы составить полный список ребер в порядке возрастания их весов. Однако полный список, вообще говоря, не требуется, так как весьма правдоподобно, что $n-1$ допустимых ребер, образующих SST, будут найдены после просмотра только «верхней» части списка, содержащей $r < m$ ребер. Отсюда немедленно следует, что процедура сортировки, используемая на шаге 2, должна быть процедурой многократного обращения, дающей корректное расположение первых p ребер в конце p-го цикла обращения. С помощью такой процедуры [34] кратчайшее ребро находится посредством одного обращения к шагу 2, затем осуществляется проверка ребра в соответствии с шагом 3; далее происходит возвращение к шагам 2 и 3 и т. д. Процесс продолжается до тех пор, пока после некоторого числа r таких проб не будут отобраны $n-1$ ребер, дающие (при добавлении их к T) SST графа. В конце такого процесса будут эффективно рассортированы только r ребер и при этом будут выполнены $r \log_2 m$ операций. Остальные $m-r$ ребер не потребуются.

Из сказанного выше следует, что, несмотря на сделанные усовершенствования, алгоритм Краскала лучше подходит для графов с небольшим числом ребер, чем для полных графов. У полных графов $m = n(n-1)/2$; для этого случая Прим [48] и Дейкстра [17] предложили алгоритмы, более эффективно использующие особенности операции J.

3.2. Алгоритм Прима [48]

Этот алгоритм порождает SST посредством разрастания только одного поддерева, например T_s, содержащего больше одной вершины. «Одиночные» вершины рассматриваются как отдельные поддеревья. Поддерево T_s постепенно разрастается за счет присоединения ребер (x_i, x_j), где $x_i \in T_s$ и $x_j \notin T_s$; причем добавляемое ребро должно иметь наименьший вес c_{ij}. Процесс продолжается до тех пор, пока число ребер в T_s не станет равным $n-1$. Тогда поддерево T_s будет требуемым SST. Впервые такая частная форма операции J была предложена Примом [48], а эффективную технику ее реализации дали Дейкстра [17] и Кевин с Уитни [35].

Алгоритм начинает работу с присвоения каждой вершине $x_j \notin T_s$ пометки $[\alpha_j, \beta_j]$, где α_j на каждом шаге есть ближайшая к x_j вершина из поддерева T_s, а β_j — вес ребра (α_j, x_j). На каждом шаге выполнения алгоритма вершина, например x_{j*}, с наименьшей пометкой β_j присоединяется к T_s посредством добавления ребра (α_{j*}, x_{j*}). Поскольку к T_s добавлена новая вершина x_{j*}, то, может быть, придется изменить пометки $[\alpha_j, \beta_j]$ у некоторых вершин $x_j \notin T_s$ (если, например, $c(x_j, x_{j*})$ меньше существующей пометки β_j) и после этого продолжить процесс. Такая процедура расстановки пометок очень похожа на ту, которая используется в задаче о кратчайшем пути при применении алгоритма Дейкстры (гл. 8, разд. 2.1).

Алгоритм имеет следующий вид:

Шаг 1. Пусть $T_s = \{x_s\}$, где x_s — произвольно выбранная вершина, и $A_s = \varnothing$ (A_s является множеством ребер, входящих в SST).

Шаг 2. Для каждой вершины $x_j \notin T_s$ найти вершину $\alpha_j \in T_s$, такую, что

$$c(\alpha_j, x_j) = \min_{x_i \in T_s} [c(x_i, x_j)] = \beta_j,$$

и приписать вершине x_j пометку $[\alpha_j, \beta_j]$. Если такой вершины α_j нет, т. е. при $\Gamma(x_j) \cap T_s = \varnothing$, приписать вершине x_j пометку $[0, \infty]$.

Шаг 3. Выбрать такую вершину x_{j*}, что

$$\beta_{j*} = \min_{x_j \notin T_s} [\beta_j].$$

Обновить данные: $T_s = T_s \cup \{x_{j*}\}$, $A_s = A_s \cup \{(\alpha_{j*}, x_{j*})\}$. Если $|T_s| = n$, то остановиться. Ребра в A_s образуют SST. Если $|T_s| \neq n$, то перейти к шагу 4.

Шаг 4. Для всех $x_j \notin T_s$, таких, что $x_j \in \Gamma(x_{j*})$, обновить пометки следующим образом.

Если $\beta_j > c(x_{j*}, x_j)$, то положить $\beta_j = c(x_{j*}, x_j)$, $\alpha_j = x_{j*}$ и вернуться к шагу 3.

Если $\beta_j \leqslant c(x_{j*}, x_j)$, то перейти к шагу 3.

3.3. Родственные задачи

До сих пор мы занимались задачей нахождения SST и в связи с этим привели описание двух алгоритмов, которые можно использовать для решения этой задачи. Применимость этих методов, однако, значительно шире, чем кажется с первого взгляда. Операция J, на которую опираются эти методы, не накладывает никаких ограничений на знак веса c_{ij} и, следовательно, описанные методы построения SST применимы также для графов с произвольными (положительными, отрицательными или нулевыми) весами ребер. Отсюда немедленно вытекает, что *длиннейшее* остовное дерево графа можно найти таким же способом, надо лишь изменить знаки весов ребер на противоположные и применить затем один из приведенных выше алгоритмов построения SST.

Более того, при доказательстве утверждения, связанного с операцией J, не используется тот факт, что полный вес остовного дерева равен сумме весов его ребер. Предполагалось только, что при замене ребра с весом C на ребро с весом $C' < C$ вес дерева уменьшается.

Таким образом, если вес дерева представляет собой монотонно возрастающую симметричную[1]) функцию, зависящую от реберных весов, то остовное дерево, минимизирующее эту весовую функцию, должно быть тем же самым SST (которое минимизирует сумму весов ребер). Например, если $C_1, C_2, \ldots, C_m$ — стоимости m ребер графа G, то остовным деревом графа G, минимизирующим $C_{i_1}^3 + C_{i_2}^3 + \ldots + C_{i_{n-1}}^3$ или $C_{i_1} \cdot C_{i_2} \cdot \ldots \cdot C_{i_{n-1}}$, где $C_{i_1}, C_{i_2}, \ldots, C_{i_{n-1}}$ — веса каких-либо $n-1$ ребер, образующих остовное дерево графа G, будет то же самое SST графа G. Добавим, что поскольку

[1]) Функция называется симметричной относительно переменных $x_1, x_2, \ldots, x_m$, если она не меняется при замене любой пары переменных друг на друга. Условия монотонности и симметричности диктуются тем, что при переходе от C к $C' < C$ вес дерева должен уменьшаться.

в первой из упомянутых выше функций третью степень можно заменить на любую другую степень $p > 0$ и поскольку при $p \to \infty$

$$\left[\sum_{l=1}^{n-1} C_{i_l}^p\right] \to \{\max_{l=1,\ldots,n-1} [C_{i_l}]\}^p,$$

то остовное дерево, у которого ребро с наибольшим весом имеет минимально возможный вес, совпадает с SST графа G.

3.4. Пример [48]

На графе G, изображенном на рис. 7.12, каждая вершина представляет некоторое лицо, а ребра (x_i, x_j) отражают тот факт, что лицо x_i может общаться с лицом x_j и наоборот. Требуется определить такой способ передачи конфиденциального сообщения между 12 лицами, при котором вероятность утечки информации будет наименьшей. Каждой передаче сообщения от x_i к x_j приписывается некоторая вероятность ρ_{ij} того, что послание может быть перехвачено посторонним лицом. Эти вероятности в процентах даны на рис. 7.12. Очевидно, что пути передачи сообщения должны образовывать остовное дерево графа G, и требуется найти такое остовное дерево, которое минимизирует величину $1 - \Pi\,(1-\rho_{ij})$, где произведение берется по тем ребрам, которые образуют это дерево. Поскольку эта функция возрастающая и симметричная относительно ρ_{ij}, то требуемое остовное дерево будет совпадать

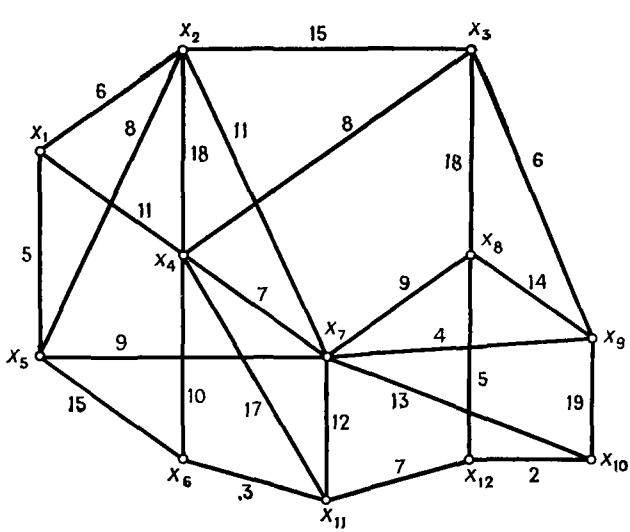


Рис. 7.12. Граф из примера 3.4.

с SST графа G, при этом ρ_{ij} принимаются за «стоимости» ребер c_{ij}.

Решим эту задачу, используя алгоритм из разд. 3.2.

Шаг 1. Возьмем $x_s = x_1$, $T_1 = \{x_1\}$, $A_1 = \varnothing$.

Шаг 2. Примем за пометки для x_2, x_4 и x_5: $[x_1, 6]$, $[x_1, 11]$ и $[x_1, 5]$ соответственно. Остальные пометки равны $[0, \infty]$.

Шаг 3. Наименьшая пометка β_j будет у вершины x_5, и поскольку $\alpha_5 = x_1$, то построим ребро (x_1, x_5): $T_1 = \{x_1, x_5\}$, $A_1 = \{(x_1, x_5)\}$.

Шаг 4. Обновим пометки вершин x_2, x_6, x_7 следующим образом:

для x_2 : $\beta_2 = 6 < c(x_5, x_2)$ и больше обновлять не надо;

для x_6 : $\beta_6 = \infty > c(x_5, x_6) = 15$, следовательно, пометка для x_6 примет вид $[x_5, 15]$;

для x_7 : $\beta_7 = \infty > c(x_5, x_7) = 9$ и пометка для x_7 примет значение $[x_5, 9]$.

Поскольку $x_4 \notin \Gamma(x_5)$, ее пометка останется такой же, как и на предыдущей итерации, т. е. $[x_1, 11]$.

Шаг 3. Пометки теперь таковы: для x_2 : $[x_1, 6]$, для x_4 : $[x_1, 11]$, для x_6 : $[x_5, 15]$, для x_7 : $[x_5, 9]$. Наименьшая пометка β_j будет у x_2, и поскольку $\alpha_2 = x_1$, то построено ребро (x_1, x_2); $T_1 = \{x_1, x_5, x_2\}$, $A_1 = \{(x_1, x_5), (x_1, x_2)\}$.

Шаг 4. Аналогично обновим пометки вершин x_3, x_4, x_7 следующим образом:

для x_3 : $[x_2, 15]$,

для x_4 : $[x_1, 11]$ (обновлять не надо),

для x_7 : $[x_5\ 9]$ (обновлять не надо).

Пометка для $x_6 \notin \Gamma(x_2)$ остается такой же, как и на предыдущей итерации, т. е. $[x_5, 15]$.

Шаг 3. Наименьшая пометка β_j будет у вершины x_7, и поскольку $\alpha_7 = x_5$, ребро (x_5, x_7) построено. $T_1 = \{x_1, x_5, x_2, x_7\}$, $A_7 = \{(x_1, x_5), (x_1, x_2), (x_5, x_7)\}$.

Шаг 4. Пометки вершин обновляются так же, как и раньше, и показаны на рис. 7.13а вместе с необходимыми для построения дерева дополнительными ребрами.

Продолжая таким образом, получим в конце SST, показанный на рис. 7.13б, с номерами ребер, указывающими, в какой последовательности они вводились в дерево.

Произведение $\Pi(1 - \rho_{ij})$ для ребер этого дерева равно 0,5214, и, следовательно, величина минимума вероятности перехвата сообщения посторонним лицом равна 47,86 %.

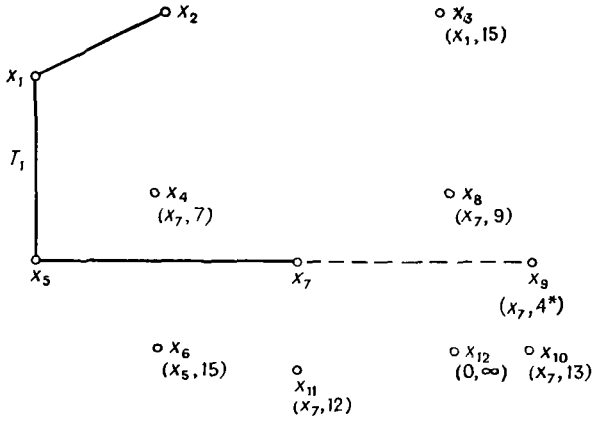


Рис. 7.13а. Частично сформированное дерево T_1 с пометками на вершинах, не принадлежащих T_1.
— — — Надо добавить следующее ребро.

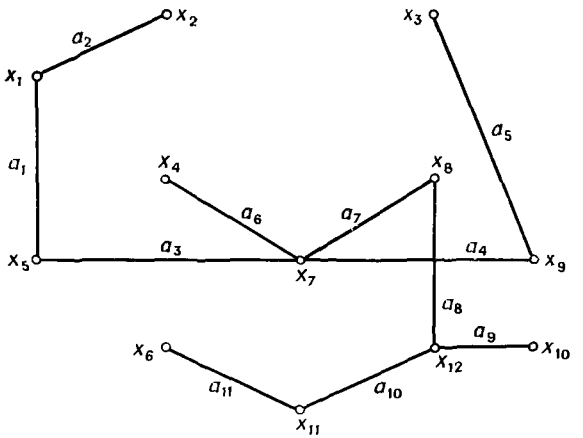


Рис. 7.13б. SST графа на рис. 7.12.

4. Задача Штейнера

В предыдущем разделе в задаче определения SST, т. е. наикратчайшего дерева графа $G = (X, \Gamma)$, ребрами «покрывались» все вершины множества X. С этой задачей тесно связана задача, известная как *«задача Штейнера на графах»* [27, 18], но последняя значительно труднее.

В этой задаче требуется найти наикратчайшее дерево T, которое «стягивает» («покрывает») заданное подмножество $P \subset X$ вер-

шин графа G. Другие вершины, принадлежащие $X - P$, могут либо стягиваться деревом T, либо нет, в зависимости от требования минимизации длины дерева T. Таким образом, задача Штейнера на графе эквивалентна нахождению наикратчайшего остовного дерева произвольного подграфа $G' = (X', \Gamma)$ графа G при условии $P \subseteq X' \subseteq X$.

Евклидова задача Штейнера впервые была поставлена как геометрическая задача [13, 14, 44, 24], в которой нужно множество точек P на евклидовой плоскости соединить линиями так, чтобы сумма длин отрезков была минимальна. Если не допускаются пересечения любых двух линий в точках вне заданного множества

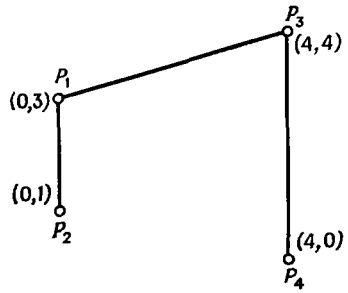


Рис. 7.14а. Кратчайшее остовное дерево. Длина = 10,123.

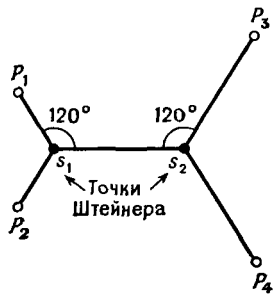


Рис. 7.14б. Кратчайшее дерево Штейнера. Длина 9,196.

P, то задача сводится к одной из задач определения SST эквивалентного графа на $|P|$ вершинах с матрицей весов, вычисленных как евклидово расстояние между точками множества P. Если допускается на плоскости введение дополнительных «искусственных» вершин (называемых точками Штейнера), то длину SST на множестве точек $P' \supset P$ можно уменьшить соответствующим подбором точек. Например, для четырех точек, показанных на рис. 7.14а, SST имеет длину, бо́льшую, чем SST графа, получаемого после добавления двух новых точек s_1 и s_2, расположенных «между» исходными точками (см. рис. 7.14б). Таким образом, для решения задачи Штейнера можно добавить в любых местах плоскости столько точек Штейнера, сколько необходимо для построения наикратчайшего дерева, стягивающего множество из P точек. Получающееся наикратчайшее дерево называют *наикратчайшим деревом Штейнера*.

Задача Штейнера на евклидовой плоскости достаточно хорошо изучена [44, 24, 11, 12], и известно большое число свойств наикратчайшего дерева Штейнера. Наиболее важными свойствами являются следующие [24, 11]:

(i) Точка Штейнера s_i имеет степень $d(s_i) = 3$. Это можно легко показать с помощью геометрического построения, поскольку угол между ребрами, инцидентными точке Штейнера s_i, должен быть равен 120° и точно три ребра инцидентны любой точке Штейнера s_i. Следовательно, эта точка является «центром» (центром Штейнера) воображаемого треугольника, вершинами которого

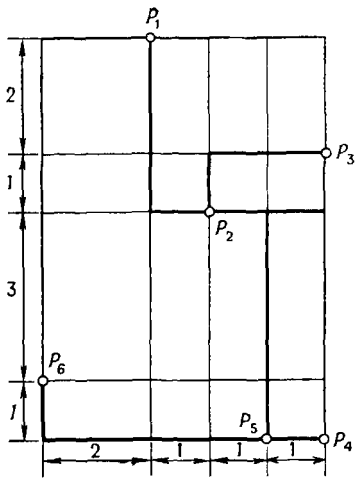


Рис. 7.15а. Кратчайшее остовное дерево. Длина = 18.

Рис. 7.15б. Кратчайшее дерево Штейнера. Длина = 15.

являются три другие точки дерева, с которыми она связана с помощью наикратчайшего дерева Штейнера.

Некоторые точки, являющиеся вершинами такого треугольника, могут оказаться другими точками Штейнера. Например, на рис. 7.14б точка Штейнера s_2 является центром воображаемого треугольника с вершинами p_3, p_4 и s_1.

(ii) Для вершины $p_i \in P$ степень $d(p_i) \leqslant 3$. Если $d(p_i) = 3$, то угол между любыми двумя ребрами, инцидентными p_i, должен быть равен 120°. Если $d(p_i) = 2$, то угол между двумя ребрами, инцидентными p_i, должен быть больше или равен 120°.

(iii) Число точек Штейнера в наикратчайшем дереве Штейнера равно k, $0 \leqslant k \leqslant n - 2$, где $n = |P|$. Доказательство упомянутых выше свойств можно найти в [24].

Несмотря на то внимание, которое уделялось евклидовой задаче Штейнера, при помощи существующих алгоритмов [44, 11] решение возможно только для небольших по размеру задач (не более 10 точек в P) и, следовательно, можно считать евклидову

задачу Штейнера нерешенной. Для задач большого размера можно обратиться к ряду эвристических методов [7, 50].

В более позднем варианте задачи Штейнера на плоскости используется обычно *линейное* (вместо евклидова) расстояние между точками. Такая постановка задач впервые была предложена Хэнном [28, 30] в связи с разработкой теории монтажа печатных схем электронных устройств. Расстояние между точками с координатами (x_1, y_1), (x_2, y_2) в этом случае определяется следующим образом:

$$d_{1,2} = |x_1 - x_2| + |y_1 - y_2|.$$

При этих условиях можно легко показать [44], что если через каждую точку из множества P провести вертикальные и горизонтальные линии, то решение задачи Штейнера можно получить, рассматривая в качестве возможных точек Штейнера точки пересечения полученной сетки линий.

Пусть граф G построен так, что множество его вершин X является множеством точек пересечения некоторой сетки линий и ребра графа G соответствуют линиям сетки, соединяющим две точки пересечения. Тогда задача Штейнера на плоскости с линейной метрикой переходит в задачу Штейнера для конечного графа, определенную в начале этого раздела [54]. На рис. 7.15б показан пример наикратчайшего дерева Штейнера для линейной задачи с шестью точками, а для сравнения на рис. 7.15а показан SST этой задачи.

Задача Штейнера для обычных неориентированных графов рассматривалась Хакими [27] и Дрейфусом и Вагнером [18]. Они нашли точные алгоритмы решения таких задач. Однако эти алгоритмы с вычислительной точки зрения являются не эффективными процедурами, хотя они и значительно лучше, чем последовательный просмотр SST всех подграфов G' графа G. В любом случае (как и в случае задач с евклидовым расстоянием) максимальный размер задач Штейнера, для решения которых требуется разумное вычислительное время, не превышает 10 вершин (в множестве P). С этой точки зрения задачу Штейнера на графе можно рассматривать как нерешенную проблему, и она в этой книге больше не будет обсуждаться.

5. Задачи

1. Доказать, что все три определения остовного дерева, данные во введении, эквивалентны.

2. Показать, что в дереве, имеющем больше одной вершины, существуют по крайней мере две вершины степени 1.

3. Показать, что детерминант любой квадратной подматрицы матрицы инциденций графа равен $+1$, -1 или 0.

4. Для матрицы размера $n \times m$ любая квадратная матрица размера $\min(n, m) \times \min(n, m)$ называется главной подматрицей. Показать, что если B_m — главная подматрица матрицы инциденций связного графа, то $|B_m|$ отличен от нуля (равен $+1$

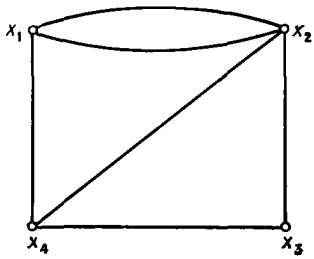


Рис. 7.16.

или -1) тогда и только тогда, когда остовный подграф, состоящий из всех ребер, соответствующих столбцам подматрицы B_m, является деревом.

5. Найти все остовные деревья графа, показанного на рис. 7.16, и, используя результат теоремы 1, убедиться, что их число равно числу, приведенному в теореме 1.

6. Показать, что произведение матриц $B_o \cdot B_o^t$, использованное в теореме 1, не обязательно вычислять перемножением B_o и B_o^t, но можно получить непосредственно из графа в виде $(n \times n)$ матрицы $M = [m_{ij}]$, определенной следующим образом: диагональный элемент m_{ii} есть степень вершины x_i, а элемент m_{ij} — число параллельных ребер между вершинами x_i и x_j со знаком минус. Чтобы получить $B_o \cdot B_0^t$, достаточно построить только $(n-1)$ строк и столбцов матрицы M.

7. Используя теорему 1, показать, что число остовов полного связного неориентированного графа с n вершинами равно n^{n-2}.

8. Пусть матрица $M = [m_{ij}]$ для ориентированного графа определена следующим образом: $m_{ii} = d_i(x_i)$, степень полузахода вершины x_i, $m_{ij} = -k$, где k есть число параллельных дуг из x_i в x_j. Показать, что ориентированный граф есть ориентированное дерево с корнем x_r тогда и только тогда, когда $m_{rr} = 0$, $m_{ii} = 1$ при $i \neq r$ и определитель подматрицы, получающейся вычеркиванием r-й строки и r-го столбца из M, равен 1 (см. [19]).

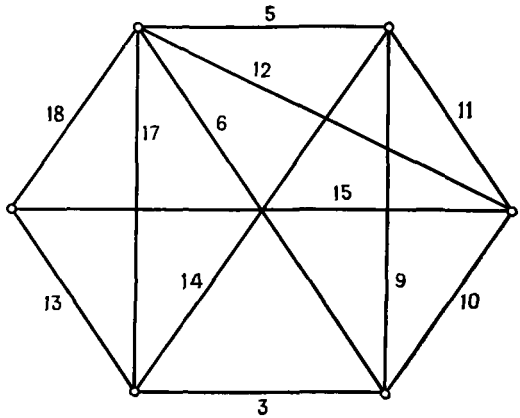


Рис. 7.17.

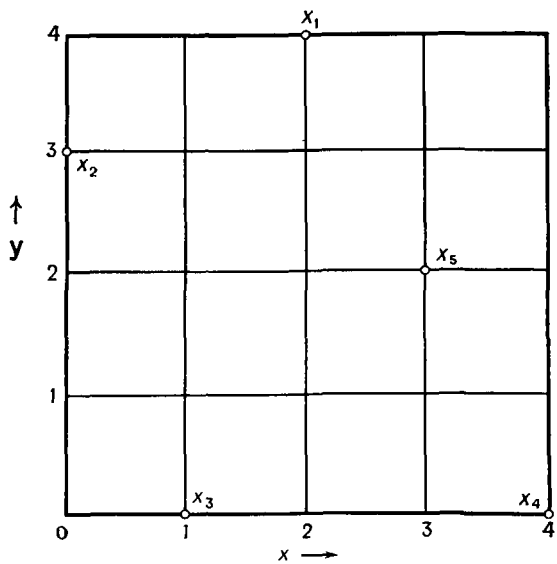


Рис. 7.18.

9. Пусть M — матрица, определенная в задаче 8. Используя полученный выше результат, показать, что число ориентированных остовов с корнем x_r в ориентированном графе (без петель) равно определителю подматрицы, полученной вычеркиванием r-й строки и r-го столбца из M (см. [1], стр. 163, и [2]).

10. Написать алгоритм перечисления всех остовов графа без повторений, используя элементарные преобразования дерева (см. [42, 47, 8]).

11. Найти SST графа, показанного на рис. 7.17, двумя алгоритмами — Крускала и Прима.

12. Найти для полного графа на множестве вершин $\{x_1, x_2, x_3, x_4\}$, как показано на рис. 7.18, с весами ребер, определенных как евклидовы расстояния:
(i) SST,
(ii) Дерево Штейнера на плоскости. (Использовать свойства точек Штейнера на плоскости, данные в разд. 4, и рассмотреть все возможные топологии (т. е. матрицы инциденций). Решить ту же задачу, полагая веса ребер равными линейным расстояниям между точками.

13. Повторить упражнение 12, добавив дополнительную вершину x_5 и оценить увеличение объема вычислений.

6. Список литературы

1. Берж К., Теория графов, М., ИЛ, 1962.
2. Bott R., Mayberry J. P. (1954), Matrices and Trees, Economic Activity Analysis, Wiley, New York.
3. Басакер Р, Саати Т., Конечные графы и сети , М., Наука, 1974.
4. Cayley A. (1874), On the mathematical theory of isomers, *Philosophical Magazine*, 67, p. 444.
5. Cayley A. (1897), Collected papers, *Quart. J. of Mathematics*, 13, Cambridge, p. 26.
6. Chan S. P., Chan S. G. (1968), Modifications of topological formulae, *IEEE Trans.*, CT—15, p. 84.
7. Change S.-K. (1972), The generation of minimal trees with Steiner topology, *J. of ACM*, 19, p. 699.
8. Chen W.-K. (1971), Applied Graph Theory, North-Holland, Amsterdam.
9. Chen W.-K. (1966), On the directed trees and directed k-trees of a digraph and their generation, *J. of SIAM* (*Appl. Math.*), 14, p. 550.
10. Chen W. K., Li H.-C. (1973), Computer generation of directed trees and complete trees, *Int. J. of Electronics*, 34, p. 1.
11. Cockayne E. J. (1970), On the efficiency of the algorithm for Steiner minimal trees, *J. of SIAM* (*Appl. Math.*), 18, p. 150.
12. Cockayne E. J., Melzak Z. A. (1968), Steiner's for problem set terminals. *Quart. Applied Mathematics*, 26, p. 213.
13. Курант Р., Роббинс Г. (1970), Что такое математика?, М., Наука.
14. Coxeter H. S. M. (1961), Introduction to geometry, Wiley, New York. [Есть русский перевод: Кокстер Г. С. М., Введение в геометрию, М., «Наука», 1966.]
15. Cummings R. L. (1966), Hamilton circuits in tree graphs, *IEEE Trans.*, CT—13, p. 82.
16. Брёйн Н. Дж. де (1968), Теория перечисления Пойа, сб. Прикладная комбинаторная математика, М., «Мир».
17. Dijkstra E. W. (1959), A note on two problems in connection with graphs. *Numerische Mathematik*, 1, p. 269.

18. Dreyfus S. E., Wagner R. A. (1972), The Steiner problem in graphs, *Networks*, 1, 195.
19. Even S. (1973), Algorithmic combinatorics, Macmillan, New York.
20. Fidler J. K., Horrocks D. H. (1973), On the generation of k-trees, *J. of Electronics*, 34, p. 185.
21. Floyd R. W. (1962), TREESORT- Algorithm 113, ACM Collected Algorithms.
22. Floyd R. W. (1964), TREESORT 3-Algorithm 245, ACM Collected Algorithms.
23. Fu Y. (1967), Application of linear graph theory to printed circuits, Proc. Asilomar Conf. on Systems and Circuits.
24. Gilbert E. N., Pollack H. O. (1968), Steiner minimal trees, *J. of SIAM (Appl. Math.)*, 16, p. 1.
25. Glover F., Klingman D. (1970), Locating stepping-stone paths in distribution problems via the predecessor index method, *Transp. Sci.*, 4, p. 220.
26. Gower J. C., Ross G. J. S. (1969), Minimum spanning trees and single linkage cluster analysis, *Applied Statistics*, 18, p. 54.
27. Hakimi S. L. (1971), Steiner's problem in graphs and its implications, *Networks*, 1, p. 113.
28. Hanan M. (1966), On Steiner's problem with rectilinear distance, *J. of SIAM (Appl. Math.)*, 14, p. 255.
29. Hanan M. (1972), A counterexample to a theorem of Fu on Steiner's problem, *IEEE Trans.*, CT—19, p. 74.
30. Hanan M., Kurzberg J. M. (1972), Placement techniques, Design automation of digital systems, Breuer, Ed., Prentice Gall, New Jersey.
31. Holzmann C. A., Harary F. (1972), On the tree graph of a matroid' *J. of SIAM (Appl. Math.)*, 22, p. 187.
32. Johnson E. (1962), Networks and basic solutions, *Ops. Res.*, 14, p. 89.
33. Kamae T. (1967), The existence of Hamiltonian circuits in tree graphs, *IEEE Trans.*, CT—14, p. 279.
34. Kershenbaum A., Van Slyke R. (1972), Computing minimum spanning trees efficiently, Proc. of the Ann. Conf. of ACM, Boston, p. 518.
35. Kevin V., Whitney M. (1972), Algorithm 422—Minimal spanning tree, *Comm. of ACM*, 15, p. 273.
36. Kirchhoff G. (1847), Annalen der Physik and Chemie, 72, p. 497.
37. Kishi G., Kajitani Y. (1968), On Hamilton circuits in tree graphs, *IEEE Trans.*, CT—15, p. 42.
38. Kishi G., Kajitani Y. (1968), On the realization of tree graphs, *IEEE Trans.*, CT—15, p. 271.
39. Knuth D. E. (1968), The art of computer programming, Vol. 1. Fundamental algorithms, Addison Wesley, Reading, Massachusetts.
40. Kruskal J. B. Jr. (1956), On the shortest spanning subtree of a graph and the traveling salesman problem, *Proc. American Mathematical Soc.*, 7, p. 48.
41. Loberman H., Weinberger A. (1957), Formal procedures for connecting terminals with a minimum total wire length, *J. of ACM*, 4, p. 428.
42. Mayeda W. (1972), Graph Theory, Wiley-Interscience, New York.
43. Mayeda W., Hakimi S. L., Chen W.-K., Deo N. (1968), Generation of complete trees, *IEEE Trans.*, CT—15, p. 101.
44 Melzak Z. A. (1961), On the problem of Steiner, *Canadian Mathematical Bulletin*, 4, p. 335.
45. Moon J. W. (1967), Various proofs of Cayley's formula for counting trees, A seminar of graph theory, Harary, Ed., Holt, Rinehart and Winston, New York.
46. Obruca A. (1964), Algorithm 1 — MINTREE, *Computer Bulletin*, p. 67.
47. Paul A. J. Jr. (1967), Generation of directed trees and 2-trees without duplication, *IEEE Trans.*, CT-14, p. 354.

48. Prim R. C. (1957), Shortest connection networks and some generalizations, *Bell Syst. Tech. J.*, **36**, p. 1389.
49. Riordan J. (1958), An Introduction to Combinatorial Analysis, Wiley, New York. [Есть русский перевод: Риордан Дж., Введение в комбинаторный анализ, М., ИЛ, 1963.]
50. Scott A. (1971), Combinatorial programming, spatial analysis and planning, Methuen, London.
51. Shank H. (1968), Note on Hamilton circuits in tree graphs, *IEEE Trans.*, **CT—15**, p. 86.
52. Srinivasan V., Thompson G. L. (1972), Accelerated algorithms for labelling and relabelling of trees with applications to distribution problems, *J. of ACM*, **19**, p. 712.
53. Trent H. M. (1954), A note on the enumeration and listing of all possible trees in a connected linear graph, *Proc. Nat. Acad. Sci. U. S. A.*, **40**, p. 1004.
54. Yang Y. Y., Wing O. (1972), Suboptimal algorithm for a wire routing problem, *IEEE Trans.*, **CT—19**, p. 508.

Глава 8

КРАТЧАЙШИЕ ПУТИ

1. Введение

Пусть дан граф $G = (X, \Gamma)$, дугам которого приписаны веса (стоимости), задаваемые матрицей $C = [c_{ij}]$. *Задача о кратчайшем пути* состоит в нахождении кратчайшего пути от заданной начальной вершины $s \in X$ до заданной конечной вершины $t \in X$, при условии, что такой путь существует, т. е. при условии $t \in R(s)$. Здесь $R(s)$ — множество, достижимое из вершины s, как это было определено в гл. 2. Элементы c_{ij} матрицы весов C могут быть положительными, отрицательными или нулями. Единственное ограничение состоит в том, чтобы в G не было циклов с суммарным отрицательным весом. Если такой цикл Ф все же существует и x_i — некоторая его вершина, то, двигаясь от s к x_i, обходя затем Ф достаточно большое число раз и попадая наконец в t, мы получим путь со сколь угодно малым ($\to -\infty$) весом. Таким образом, в этом случае кратчайшего пути не существует.

Если, с другой стороны, такие циклы существуют, но исключаются из рассмотрения, то нахождение кратчайшего пути (простой цепи) между s и t эквивалентно нахождению в этом графе кратчайшего гамильтонова пути с концевыми вершинами s и t. Это можно усмотреть из следующего факта. Если из каждого элемента c_{ij} матрицы весов C вычесть достаточно большое число L, то получится новая матрица весов $C' = [c'_{ij}]$, все элементы c'_{ij} которой отрицательны. Тогда кратчайший путь от s к t — с исключением отрицательных циклов — необходимо будет гамильтоновым, т. е. проходящим через все другие вершины. Так как вес любого гамильтонова пути с матрицей весов C' равен весу этого пути с матрицей C, но уменьшенному на постоянную величину $(n-1)\cdot L$, то кратчайший путь (простая цепь) от s к t с матрицей C' будет кратчайшим гамильтоновым путем от s к t при первоначальной матрице C. Задача нахождения кратчайшего гамильтонова пути намного сложнее, чем задача о кратчайшем пути; она обсуждается отдельно в гл. 10. Поэтому мы будем предполагать, что все циклы в G имеют неотрицательный суммарный вес. Отсюда также вытекает, что неориентированные дуги (ребра) графа G не могут иметь отрицательные веса.

Следующие задачи являются непосредственными обобщениями сформулированной выше задачи о кратчайшем пути.

(i) Для заданной начальной вершины s найти кратчайшие пути между s и *всеми* другими вершинами $x_i \in X$.

(ii) Найти кратчайшие пути между *всеми парами* вершин.

В последующих разделах будет показано, что почти все методы, позволяющие решить задачу о кратчайшем $(s\text{-}t)$-пути, дают также (в процессе решения) и все кратчайшие пути от s к x_i $(Ax_i \in X)$. Таким образом, они позволяют решить задачу с небольшими дополнительными вычислительными затратами. С другой стороны, задача (i) может быть решена либо n-кратным применением алгоритма задачи (ii), причем на каждом шаге в качестве начальной вершины s берутся различные вершины, либо однократным применением специального алгоритма.

В настоящей главе мы дадим общие алгоритмы решения сформулированных выше задач и частные алгоритмы для случаев, когда все c_{ij} неотрицательны. Эти частные случаи встречаются на практике довольно часто (например, когда c_{ij} являются расстояниями), так что рассмотрение этих специальных алгоритмов оправдано. Мы будем предполагать, что матрица не удовлетворяет, вообще говоря, условию треугольника, т. е. не обязательно $c_{ij} \leqslant c_{ik} + c_{kj}$ для всех i, j и k. В противном случае кратчайший путь между x_i и x_j состоит из одной единственной дуги [1]) $(x_i\ x_j)$ и задача становится тривиальной. Если в графе G дуга (x_i, x_j) отсутствует, то ее вес полагается равным ∞.

На практике часто требуется найти не только кратчайший путь, но также второй, третий и т. д. кратчайшие пути в графе. Располагая этими результатами, можно решить, какой путь выбрать в качестве наилучшего (указанный подход полезен при использовании таких критериев, которые являются субъективными по своей природе или не могут быть непосредственно включены в алгоритм). Кроме того, второй, третий и т. д. кратчайшие пути можно использовать при анализе «чувствительности» задачи о кратчайшем пути. В этой главе мы опишем недавно полученный алгоритм вычисления K кратчайших простых цепей между двумя заданными вершинами общего графа.

В настоящей главе обсуждаются также задачи нахождения в графах путей с максимальной надежностью и с максимальной пропускной способностью. Эти задачи связаны с задачей о кратчайшем пути, хотя в них характеристика пути (скажем, вес) является не суммой, а некоторой другой функцией характеристик (весов) дуг, образующих путь. Такие задачи можно переформулировать как задачи о кратчайшем пути. Однако можно поступить

---

[1]) Если такая дуга существует — *Прим. ред.*

иначе: непосредственно приспособить для их решения те методы, которые применяются в задачах о кратчайшем пути.

Обсуждается также случай, когда учитываются и пропускные способности, и надежности дуг. Это приводит к задаче о пути с наибольшей ожидаемой пропускной способностью. И хотя такая частная задача не может быть решена при помощи техники отыскания кратчайшего пути, но итерационный алгоритм, использующий эту технику в качестве основного шага, является эффективным методом получения оптимального ответа.

Задачи о кратчайшем пути, в которых на пути накладываются некоторые ограничения (см. [4], [12], [23]), в настоящей главе не рассматриваются, так как к ним непосредственно не применимы те методы расстановки пометок, на которых базируются все описанные здесь алгоритмы нахождения кратчайшего пути. Эти задачи с ограничениями часто настолько сложны, что соответствующие алгоритмы позволяют находить оптимальные решения лишь таких задач, размеры которых (например, число вершин) на несколько порядков меньше, чем у аналогичных задач без ограничений. Ряд важных задач с ограничениями, например задача о кратчайшем гамильтоновом пути, рассматривается в отдельных главах

2. Кратчайший путь между двумя заданными вершинами s и t

Сначала мы приведем очень простой и эффективный алгоритм решения этой задачи для случая $c_{ij} \geqslant 0$ $(\forall i,j)$, а затем распространим описанный метод на общий случай $c_{ij} \gtrless 0$ с оговоркой, что циклы с отрицательными весами отсутствуют.

2.1. Случай неотрицательной матрицы весов

Наиболее эффективный алгоритм решения задачи о кратчайшем (s-t)-пути первоначально дал Дейкстра [10]. В общем случае этот метод основан на приписывании вершинам временных пометок, причем пометка вершины дает верхнюю границу длины пути от s к этой вершине. Эти пометки (их величины) постепенно уменьшаются с помощью некоторой итерационной процедуры, и на каждом шаге итерации точно одна из временных пометок становится постоянной. Последнее указывает на то, что пометка уже не является верхней границей, а дает точную длину кратчайшего пути от s к рассматриваемой вершине. Опишем этот метод подробно.

2.1.1. Алгоритм Дейкстры ($c_{ij} \geqslant 0$)

Пусть $l(x_i)$ — пометка вершины x_i.

Присвоение начальных значений

Шаг 1. Положить $l(s) = 0$ и считать эту пометку постоянной. Положить $l(x_i) = \infty$ для всех $x_i \neq s$ и считать эти пометки временными. Положить $p = s$.

Обновление пометок

Шаг 2. Для всех $x_i \in \Gamma(p)$, пометки которых временные, изменить пометки в соответствии со следующим выражением:

$$l(x_i) \leftarrow \min [l(x_i),\ l(p) + c(p,\ x_i)]. \tag{8.1}$$

Превращение пометки в постоянную

Шаг 3. Среди всех вершин с временными пометками найти такую, для которой $l(x_i^*) = \min [l(x_i)]$.

Шаг 4. Считать пометку вершины x_i^* постоянной и положить $p = x_i^*$.

Шаг 5. (i) (*Если надо найти лишь путь от s к t.*) Если $p = t$, то $l(p)$ является длиной кратчайшего пути.
Останов.
Если $p \neq t$, перейти к шагу 2.

(ii) (*Если требуется найти пути от s ко всем остальным вершинам.*)
Если все вершины отмечены как постоянные, то эти пометки дают длины кратчайших путей. Останов.
Если некоторые пометки являются временными, перейти к шагу 2.

Доказательство того, что вышеприведенный алгоритм действительно дает кратчайшие пути, чрезвычайно простое, дадим набросок этого доказательства.

Допустим, что на некотором этапе постоянные пометки дают длины кратчайших путей. Пусть S_1 — множество вершин с этими пометками, а S_2 — множество вершин с временными пометками. В конце шага 2 каждой итерации временная пометка $l(x_i)$ дает кратчайший путь от s к x_i, проходящий полностью по вершинам множества S_1. (Так как при каждой итерации в множество S_1 включается только одна вершина, то обновление пометки $l(x_i)$ требует только одного сравнения на шаге 2.)

Пусть кратчайший путь от s к x_i^* не проходит целиком по S_1 и содержит по крайней мере одну вершину из S_2, и пусть $x_j \in S_2$ — первая такая вершина в этом пути. Так как по предположению c_{ij} неотрицательны, то часть пути от x_j к x_i^* должна иметь неотрицательный вес Δ и $l(x_j) < l(x_j^*) - \Delta < \Delta(x_i^*)$. Это, однако, противоречит утверждению, что $l(x_i^*)$ — наименьшая временная пометка, и, следовательно, кратчайший путь к x_i^* проходит полностью по вершинам множества S_1, и поэтому $l(x_i^*)$ является его длиной.

Так как вначале множество S_1 равно $\{s\}$ и при каждой итерации к S_1 добавляется x_i^*, то предположение, что $l(x_i)$ равно длине кратчайшего пути $\forall\, x_i \in S_1$, выполняется при каждой итерации. Отсюда по индукции следует, что алгоритм дает оптимальный ответ.

Если требуется найти кратчайшие пути между s и всеми другими вершинами полного связного графа с n вершинами, то в процессе работы алгоритма выполняются $n(n-1)/2$ операций сложения и сравнения на шаге 2 и еще $n(n-1)/2$ операций сравнения на шаге 3. Кроме того, при осуществлении шагов 2 и 3 необходимо определить, какие вершины являются временными, а для этого нужно еще $n(n-1)/2$ операций сравнения. Эти величины являются верхними границами для числа операций, необходимых при отыскании кратчайшего пути между заданными вершинами s и t. Они действительно достигаются, если окажется, что вершина t будет последней вершиной, получившей постоянную пометку. (В [22] Джонсон предложил так называемый метод сортировки, позволяющий уменьшить число операций на шаге 3.)

Как только длины кратчайших путей от s будут найдены (они будут заключительными значениями пометок вершин), сами пути можно получить при помощи рекурсивной процедуры с использованием соотношения (8.2). Так как вершина x_i' непосредственно предшествует вершине x_i в кратчайшем пути от s к x_i, то для любой вершины x_i соответствующую вершину x_i' можно найти как одну из оставшихся вершин, для которой

$$l(x_i') + c(x_i', x_i) = l(x_i). \tag{8.2}$$

Если кратчайший путь от s до любой вершины x_i является единственным, то дуги (x_i', x_i) этого кратчайшего пути образуют ориентированное дерево (см. предыдущую главу) с корнем s. Если существует несколько «кратчайших» путей от s к какой-либо другой вершине, то при некоторой фиксированной вершине x_i' соотношение (8.2) будет выполняться для более чем одной вершины x_i. В этом случае выбор может быть либо произвольным (если нужен какой-то один кратчайший путь между s и x_i), либо таким, что рассматриваются все дуги (x_i', x_i), входящие в какой-либо из кратчайших путей, и при этом совокупность всех таких дуг образует не ориентированное дерево, а общий граф, называемый *базой относительно* s или кратко — *s-базой* [1]).

2.1.2. Пример. Рассмотрим граф, изображенный на рис. 8.1, где каждое неориентированное ребро рассматривается как пара противоположно ориентированных дуг равного веса. Матрица

[1]) Заметим, что нет никакой связи между понятием базы и понятием базы графа, введенным ранее в гл. 2.

весов C_1 приведена ниже. Требуется найти все кратчайшие пути от вершины x_1 ко всем остальным вершинам. Мы воспользуемся

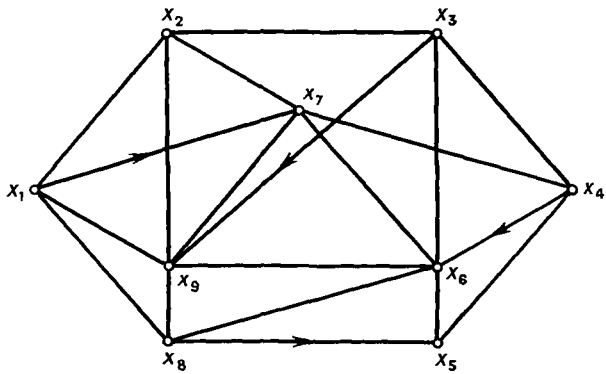


Рис. 8.1. Граф из примера 2.1.2.

алгоритмом Дейкстры. Постоянные пометки будем снабжать знаком $+$, остальные пометки рассматриваются как временные.

C_1	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
x_1		10					3	6	12
x_2	10		18				2		13
x_3		18		25		20			7
x_4			25		5	16	4		
x_5				5		10			
x_6			20		10		14	15	9
x_7		2		4		14			24
x_8	6				23	15			5
x_9	12	13				9	24	5	

Алгоритм работает так:

Шаг 1. $l(x_1) = 0^+$, $l(x_i) = \infty \ \forall x_i \neq x_1$, $p = x_1$.

Первая итерация

Шаг 2. $\Gamma(p)=\Gamma(x_1)=\{x_2, x_7, x_8, x_9\}$ — все пометки временные. Возьмем сначала x_2. Из (8.1) получаем

$$l(x_2)=\min[\infty, 0^+ + 10]=10,$$

аналогично $l(x_7)=3$, $l(x_8)=6$, $l(x_9)=12$.

Шаг 3. $\min(\underbrace{10}_{x_2}, \underbrace{3}_{x_7}, \underbrace{6}_{x_8}, \underbrace{12}_{x_9}, \underbrace{\infty}_{x_3,x_4,x_5,x_6}) = 3$ соответствует x_7.

Шаг 4. x_7 получает постоянную пометку $\boxed{l(x_7)=3^+}$, $p=x_7$.

Шаг 5. Не все вершины имеют постоянные пометки, поэтому переходим к шагу 2. Пометки в начале следующей итерации показаны на рис. 8.2 (а).

Вторая итерация

Шаг 2. $\Gamma(p)=\Gamma(x_7)=\{x_2, x_4, x_6, x_9\}$ — все пометки временные. Из соотношения (8.1) имеем

$$l(x_2)=\min[10, 3^+ + 2]=5,$$

аналогично $l(x_4)=7$, $l(x_6)=17$, $l(x_9)=12$. Пометки изображены на рис. 8.2 (б).

Шаг 3. $\min[\underbrace{5}_{x_2}, \underbrace{7}_{x_4}, \underbrace{17}_{x_6}, \underbrace{6}_{x_8}, \underbrace{12}_{x_9}, \underbrace{\infty}_{x_3,x_5}] = 5$ соответствует x_2.

Шаг 4. x_2 получает постоянную пометку $\boxed{l(x_2)=5^+}$, $p=x_2$.

Шаг 5. Перейти к шагу 2.

Третья итерация

Шаг 2. $\Gamma(p)=\Gamma(x_2)=\{x_1, x_3, x_7, x_9\}$ — только вершины x_3 и x_9 имеют временные пометки; из соотношения (8.1) получаем

$$l(x_3)=\min[\infty, 5^+ + 18]=23$$

и аналогично

$$l(x_9)=12.$$

Шаг 3. $\min[\underbrace{23}_{x_3}, \underbrace{7}_{x_4}, \underbrace{17}_{x_6}, \underbrace{6}_{x_8}, \underbrace{12}_{x_9}, \underbrace{\infty}_{x_5}] = 6$ соответствует x_8.

Шаг 4. x_8 получает постоянную пометку $\boxed{l(x_8)=6^+}$, $p=x_8$

Шаг 5. Перейти к шагу 2.

Продолжая этот процесс, получим окончательную картину расстановки пометок, изображенную на рис. 8.2 (в). Для нахождения кратчайшего пути между вершиной x_2 (например) и начальной вершиной x_1 мы последовательно используем соотношение (8.2). Таким образом, полагая $x_i=x_2$, находим вершину x_2', непо-

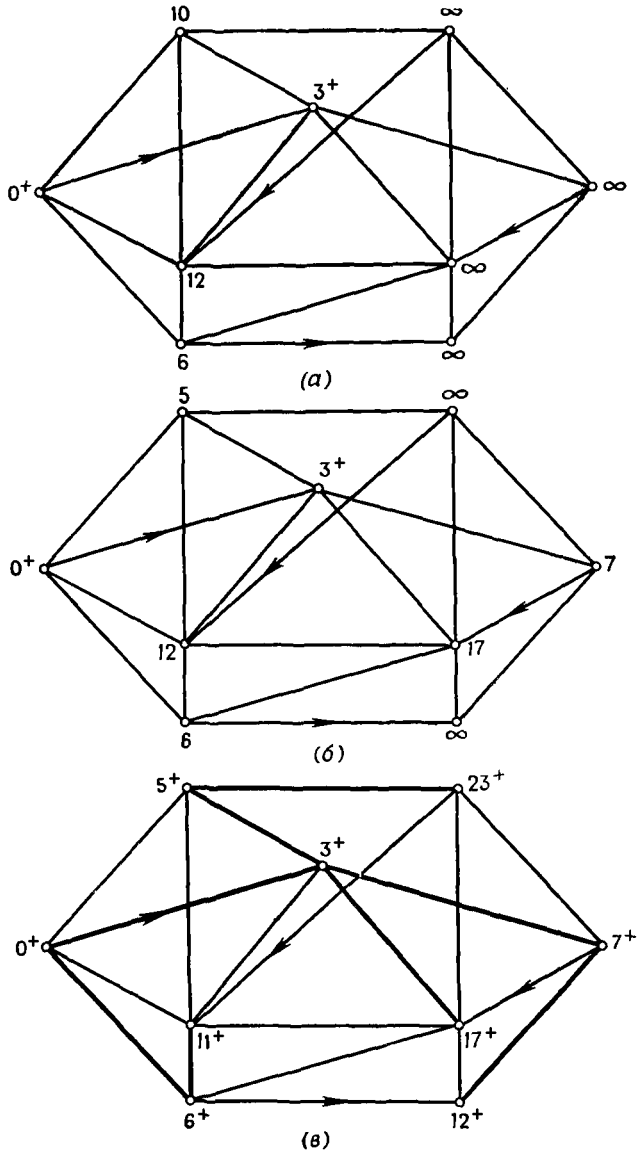


Рис. 8.2. (а) Пометки в конце 1-й итерации. (б) Пометки в конце шага 2 на 2-й итерации. (в) Окончательные пометки вершин и x_1-база.

средственно предшествующую x_2 в кратчайшем пути от x_1 к x_2: вершина x_2' должна удовлетворять соотношению

$$l(x_2') + c(x_2', x_2) = l(x_2) = 5.$$

Единственной такой вершиной является x_7. Далее, применяем второй раз соотношение (8.2), беря $x_i = x_7$; получаем вершину x_7', непосредственно предшествующую x_7 в кратчайшем пути от x_1 к x_2. Вершина x_7' удовлетворяет соотношению

$$l(x_7') + c(x_7', x_7) = l(x_7) = 3.$$

Единственной такой вершиной является x_1, и поэтому кратчайший путь от x_1 к x_2 есть (x_1, x_7, x_2). x_1-база, дающая все кратчайшие пути от x_1, представляет собой дерево, изображенное жирными линиями на рис. 8.2 (в).

2.2. Случай общей матрицы весов

Только что описанный алгоритм Дейкстры применим лишь в том случае, когда $c_{ij} \geqslant 0$ для всех i и j. Однако если матрица C является матрицей стоимостей, то дуги, приносящие доход, должны иметь отрицательные «стоимости». В этом случае для нахождения кратчайших путей между вершиной s и всеми другими вершинами можно воспользоваться описанной ниже процедурой. Этот метод также является итерационным и основан на пометках вершин, причем в конце k-й итерации пометки равны длинам тех кратчайших путей (от s ко всем остальным вершинам), которые содержат не более $k + 1$ дуг. В отличие от алгоритма Дейкстры никакая из пометок во время этого процесса не рассматривается как окончательная. Описываемый метод был первоначально предложен в середине 50-х годов Фордом [14], Муром [26] и Беллманом [2]. Перейдем к его описанию.

2.2.1. Алгоритм для общей матрицы весов

Пусть $l^k(x_i)$ — пометка вершины x_i в конце $(k+1)$-й итерации.

Присвоение начальных знаний

Шаг 1. Положим $S = \Gamma(s)$, $k = 1$, $l^1(s) = 0$, $l^1(x_i) = c(s, x_i)$ для всех $x_i \in \Gamma(s)$ и $l^1(x_i) = \infty$ для всех остальных x_i.

Обновление пометок

Шаг 2. Для каждой вершины $x_i \in \Gamma(s)$ $(x_i \neq s)$ изменить ее пометку следующим образом:

$$l^{k+1}(x_i) = \min\left[l^k(x_i), \min_{x_j \in T_i}\{l^k(x_j) + c(x_j, x_i)\}\right], \qquad (8.3)$$

где $T_i = \Gamma^{-1}(x_i) \cap S$. (Множество S содержит теперь все вершины, для которых кратчайшие пути из s состоят из k дуг.)

Множество T_i содержит те вершины, для которых текущие кратчайшие пути из s состоят из k дуг (т. е. те, вершины которых лежат в S) и для которых существуют дуги к вершине x_i. Отметим, что если $x_i \notin \Gamma(S)$, то кратчайший путь от s к x_i не может состоять из $k+1$ дуг и поэтому изменять пометку вершины x_i не нужно. Для вершин $x_i \notin \Gamma(S)$ положим $l^{k+1}(x_i) = l^k(x_i)$.

Проверка на окончание

Шаг 3. (а) Если $k \leqslant n-1$ и $l^{k+1}(x_i) = l^k(x_i)$ для всех x_i, то получен оптимальный ответ и пометки равны длинам кратчайших путей. Останов.

(б) Если $k < n-1$ и $l^{k+1}(x_i) \neq l^k(x_i)$ для некоторой вершины x_i, то перейти к шагу 4.

(в) Если $k = n-1$ и $l^{k+1}(x_i) \neq l^k(x_i)$ для некоторой вершины x_i, то в графе существует цикл отрицательного веса и задача не имеет решения. Останов.

Подготовка к следующей итерации

Шаг 4. Обновить множество следующим образом:

$$S = \{x_i \mid l^{k+1}(x_i) \neq l^k(x_i)\}. \tag{8.4}$$

(Новое множество S содержит теперь все вершины, кратчайшие пути до которых из s имеют длину $k+1$.)

Шаг 5. Положить $k = k+1$ и перейти к шагу 2.

Как только получены длины кратчайших путей от s к каждой другой вершине, то сами пути опять находятся просто, с помощью соотношения (8.2). Пути могут быть получены и другим способом, если в дополнение к пометке $l^k(x_i)$ для каждой вершины хранить во время вычислений другую пометку $\theta^k(x_i)$. Пометка $\theta^k(x_i)$ указывает вершину, непосредственно предшествующую вершине x_i в кратчайшем пути от s к x_i во время k-й итерации. Можно начать с $\theta^1(x_i) = s \; \forall \; x_i \in \Gamma(s)$ и для всех остальных вершин x_i выбрать $\theta^1(x_i)$ произвольно (скажем, равной 0). Пометки $\theta^k(x_i)$ можно тогда изменять в соответствии с соотношением (8.3). Таким образом, $\theta^{k+1}(x_i) = \theta^k(x_i)$, если в квадратных скобках в выражении (8.3) будет наименьшим первый член, или $\theta^{k+1}(x_i) = x_j$, если наименьшим является второй член. Если $\theta(x_i)$ — вектор, составленный из пометок θ при завершении работы алгоритма, то кратчайший путь от s к x_i получается в обратном порядке, а именно $s, \ldots, \theta^3(x_i), \theta^2(x_i), \theta(x_i), x_i$, где $\theta^2(x_i)$ является сокращением для $\theta(\theta(x_i))$ и т. д.

Доказательство того, что приведенный алгоритм дает оптимальное решение, весьма простое. Здесь мы его не приводим. Одна-

ко заметим, что оно базируется на принципе динамического программирования и том факте, что если не существует никакого оптимального пути из k дуг, то не может также существовать и никакого оптимального пути, содержащего $k+1$ дуг [4]. Описанный алгоритм можно применять и в случае неотрицательной матрицы весов, хотя это, вообще говоря, намного хуже, чем использование алгоритма Дейкстры. В случае полного связного графа с n вершинами этот алгоритм требует порядка n^3 операций сложения и сравнения, в то время как в алгоритме Дейкстры требуется n^2 операций. Некоторые улучшения описанного алгоритма, предложенные Йеном, позволяют уменьшить число операций в четыре раза, но порядок роста остается равным трем.

2.2.2. Пример. Рассмотрим граф, изображенный на рис. 8.3, где опять неориентированные ребра рассматриваются как пары

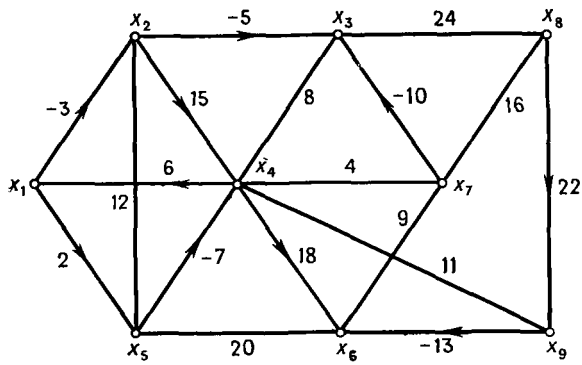


Рис. 8.3. Граф из примера 2.2.2.

противоположно ориентированных дуг с равными весами. Веса указаны у соответствующих дуг и могут быть как положительными, так и отрицательными числами. Требуется найти кратчайшие пути от x_1 ко всем остальным вершинам, при условии, что граф не содержит циклов отрицательного веса, или найти такой цикл, если он существует.

Алгоритм работает следующим образом.

Присвоение начальных значений

Шаг 1. $s = x_1$, $S = \{x_2, x_5\}$, $l^1(x_1) = 0$, $l^1(x_2) = -3$, $l^1(x_5) = 2$, $l^1(x_i) = \infty$ для всех других x_i. Положить $k = 1$.

Первая итерация

Шаг 2. $\Gamma(S) = \{x_2, x_3, x_4, x_5, x_6\}$. Поэтому для x_2 : $T_2 = \{x_1, x_5\} \cap \{x_2, x_5\} = \{x_5\}$ и из соотношения (8.3) получаем

$$l^2(x_2) = \min[-3, \underbrace{\{l^1(x_5) + c(x_5, x_2)\}}_{x_j = x_5}] =$$

$$= \min[-3, (2+12)] =$$

$$= -3;$$

для x_3: $T_3 = \{x_2, x_4, x_7, x_8\} \cap \{x_2, x_5\} = \{x_2\}$,

$$l^2(x_3) = \min[\infty, \underbrace{(-3-5)}_{x_j = x_2}] = -8;$$

для x_4: $T_4 = \{x_2, x_3, x_5, x_7, x_9\} \cap \{x_2, x_5\} = \{x_2, x_5\}$,

$$l^2(x_4) = \min[\infty, \min\{\underbrace{(-3+15)}_{x_j = x_2}, \underbrace{(2-7)}_{x_j = x_5}\}] = -5;$$

для x_5: $T_5 = \{x_1, x_2\, x_6\} \cap \{x_2, x_5\} = \{x_2\}$,

$$l^2(x_5) = \min[2, \underbrace{(-3+12)}_{x_j = x_2}] = 2;$$

для x_6: $T_6 = \{x_4, x_5, x_7, x_9\} \cap \{x_2, x_5\} = \{x_5\}$,

$$l^2(x_6) = \min[\infty, \underbrace{(2+20)}_{x_j = x_5}] = 22.$$

Пометки $l^2(x_i)$ таковы: $[0, -3, -8, -5, 2, 22, \infty, \infty, \infty]$ для $x_i = x_1, x_2, \ldots, x_9$ соответственно.

Шаг 3 (б). Перейти к шагу 4.

Шаг 4. $S = \{x_3, x_4, x_6\}$.

Шаг 5. $k = 2$, перейти к шагу 2.

Вторая итерация

Шаг 2. $\Gamma(S) = \{x_1, x_3, x_4, x_5, x_6, x_7, x_8, x_9\}$;

для x_3: $T_3 = \{x_2, x_4, x_7, x_8\} \cap \{x_3, x_4, x_6\} = \{x_4\}$,

$$l^3(x_3) = \min[-8, \underbrace{(-5+8)}_{x_j = x_4}] = -8;$$

для x_4: $T_4 = \{x_2, x_3, x_5, x_7, x_9\} \cap \{x_3, x_4, x_6\} = \{x_3\}$,

$$l^3(x_4) = \min[-5, \underbrace{(-8+8)}_{x_j = x_3}] = -5;$$

для x_5: $T_5 = \{x_1, x_2, x_6\} \cap \{x_3, x_4, x_6\} = \{x_6\}$,
$l^3(x_5) = \min[2, \underbrace{(22+20)}_{x_j=x_6}] = 2$;

для x_6: $T_6 = \{x_4, x_5, x_7, x_9\} \cap \{x_3, x_4, x_6\} = \{x_4\}$,
$l^3(x_6) = \min[22, \underbrace{(-5+18)}_{x_j=x_4}] = 13$;

для x_7: $T_7 = \{x_4, x_6, x_8\} \cap \{x_3, x_4, x_6\} = \{x_4, x_6\}$,
$l^3(x_7) = \min[\infty, \min\{\underbrace{(-5+4)}_{x_j=x_4}, \underbrace{(22+9)}_{x_j=x_6}\}] = -1$;

для x_8: $T_8 = \{x_3, x_7\} \cap \{x_3, x_4, x_6\} = \{x_3\}$,
$l^3(x_8) = \min[\infty, \underbrace{(-8+24)}_{x_j+x_3}] = 16$;

для x_9: $T_9 = \{x_4, x_8\} \cap \{x_3, x_4, x_6\} = \{x_4\}$,
$l^6(x_9) = \min[\infty, \underbrace{(-5+11)}_{x_j=x_4}] = 6$.

Пометки $l^3(x_i)$ равны [0, —3, —8, —5, 2, 13, —1, 16, 6] для $x_i = x_1, x_2, \ldots, x_9$ соответственно.

Шаг 3 (*б*). Перейти к шагу 4.

Шаг 4. $S = \{x_6, x_7, x_8, x_9\}$.

Шаг 5. $k = 3$, перейти к шагу 2.

И т. д.

Продолжая этот процесс, получаем результаты, приведенные ниже в краткой форме.

Третья итерация

Шаг 2. $\Gamma(S) = \{x_3, x_4, x_5, x_6, x_7, x_8, x_9\}$,
$T_3 = \{x_7, x_8\}$, $l^4(x_3) = -11$,
$T_4 = \{x_7, x_9\}$, $l^4(x_4) = -5$,
$T_5 = \{x_6\}$, $l^4(x_5) = 2$,
$T_6 = \{x_7, x_9\}$, $l^4(x_6) = -7$,
$T_7 = \{x_6, x_8\}$, $l^4(x_7) = -1$,
$T_8 = \{x_7\}$, $l^4(x_8) = 15$,
$T_9 = \{x_8\}$, $l^4(x_9) = 6$.

Вектор пометок $l^4(x_i)$ равен [0, —3, —11, —5, 2, —7, —1 15, 6.]

Шаг 4. $S = \{x_3, x_6, x_8\}$.

Четвертая итерация

Шаг. 2. $\Gamma(S) = \{x_3, x_4, x_5, x_7, x_8, x_9\}$,

$T_3 = \{x_8\}$, $l^5(x_3) = -11$,
$T_4 = \{x_3\}$, $l^5(x_4) = -5$,
$T_5 = \{x_6\}$, $l^5(x_5) = 2$,
$T_7 = \{x_6, x_8\}$, $l^5(x_7) = -1$,
$T_8 = \{x_3\}$, $l^5(x_8) = 13$,
$T_9 = \{x_8\}$, $l^5(x_9) = 6$.

Вектор пометок $l^5(x_i)$ равен [0, —3, —11, —5, 2, —7, —1, 13, 6].

Шаг 4. $S = \{x_8\}$.

Пятая итерация

Шаг 2. $\Gamma(S) = \{x_3, x_7, x_9\}$,

$T_3 = \{x_8\}$, $l^6(x_3) = -11$,
$T_7 = \{x_8\}$, $l^6(x_7) = -1$,
$T_9 = \{x_9\}$, $l^6(x_9) = 6$.

Шаг 3 (а). Останов.

Вектор пометок $l^6(x_i)$ совпадает с $l^5(x_i)$, и, следовательно, эти пометки равны длинам кратчайших путей. Сами пути строятся

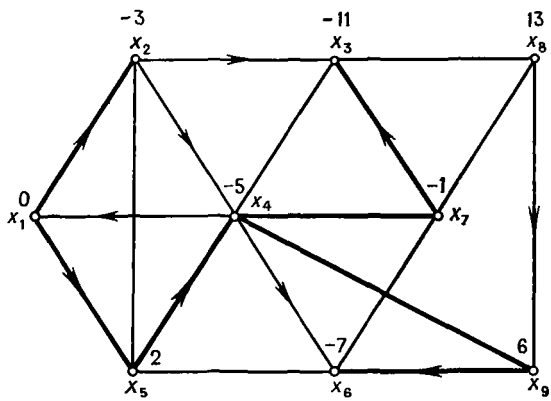


Рис. 8.4. Окончательные пометки вершин и x_1-база.

последовательно, как и в том случае, когда использовалось соотношение (8.2). Соответствующая x_1-база показана жирными линиями на рис. 8.4.

3. Кратчайшие пути между всеми парами вершин

Пусть требуется найти кратчайшие пути между всеми парами вершин графа. Очевидный способ получить ответ на этот вопрос заключается в n-кратном применении алгоритма предыдущего раздела, причем каждый раз в качестве начальной вершины s берутся различные вершины. В случае полного графа с неотрицательной матрицей весов C время, необходимое для вычислений, пропорционально n^3, а для произвольной матрицы весов оно пропорционально n^4. Поэтому если задача о кратчайшем пути имеет большую размерность, то ее невозможно решить с помощью последовательного применения алгоритма из разд. 2.2.

В настоящем разделе мы опишем совершенно иной подход к задаче нахождения кратчайших путей между всеми парами вершин. Этот метод применим как к неотрицательным, так и к произвольным матрицам весов и время, необходимое для вычислений, пропорционально n^3. Если этот метод применить к графам с неотрицательной матрицей весов, то он сэкономит почти 50% времени [11] по сравнению с n-кратным применением алгоритма Дейкстры Метод был предложен первоначально Флойдом [13] и развит Мерчлэндом [27]. Он базируется на использовании последовательности из n преобразований (итераций) начальной матрицы весов C При этом на k-й итерации матрица представляет длины кратчайших путей между каждой парой вершин с тем ограничением, что путь между x_i и x_j (для любых x_i и x_j) содержит в качестве промежуточных только вершины из множества $\{x_1, x_2, \ldots, x_k\}$.

3.1. Алгоритм Флойда (для произвольной матрицы весов)

Предположим, что в начальной матрице весов $c_{ii} = 0$ для всех $i = 1, 2, \ldots, n$ и $c_{ij} = \infty$, если в графе отсутствует дуга (x_i, x_j).

Присвоение начальных значений

Шаг 1. Положить $k = 0$.

Итерация

Шаг 2. $k = k + 1$.

Шаг 3. Для всех $i \neq k$, таких, что $c_{ik} \neq \infty$, и для всех $j \neq k$, таких, что $c_{kj} \neq \infty$, введем операцию

$$c_{ij} = \min [c_{ij}, (c_{ik} + c_{kj})]. \tag{8.5}$$

Проверка на окончание

Шаг 4. (а) Если $c_{ii} < 0$, то в графе G существует цикл отрицательного веса, содержащий вершину x_i, и решения нет. Останов.

(б) Если все $c_{ii} \geqslant 0$ и $k = n$, то получено решение. Матрица $[c_{ij}]$ дает длины всех кратчайших путей. Останов.

(в) Если все $c_{ii} \geqslant 0$, но $k < n$, то вернуться к шагу 2.

Доказательство оптимальности ответа, полученного с помощью этого алгоритма, чрезвычайно простое [21], [27], и мы предоставляем его читателю. Основная операция алгоритма, определяемая соотношением (8.5), называется *трехместной операцией*. Она имеет разнообразные применения в задачах той же природы, что и задача о кратчайшем пути. Такие задачи обсуждаются в последующих разделах.

Сами кратчайшие пути можно найти по их длинам с помощью рекурсивной процедуры, подобной той, которая выше определялась соотношением (8.2). С другой стороны, можно использовать технику, предложенную Ху [21], для записи информации о самих путях (наряду с информацией о длинах путей). Этот последний метод аналогичен использованному в разд. 2.2.1 и особенно полезен в тех случаях, когда требуется найти в графе цикл отрицательного веса (если такой существует). В этом методе в дополнение к матрице весов C хранится и обновляется вторая $(n \times n)$-матрица $\mathbf{\Theta} = [\theta_{ij}]$. Элемент θ_{ij} указывает вершину, непосредственно предшествующую вершине x_j в кратчайшем пути от x_i к x_j. Матрице $\mathbf{\Theta}$ присваиваются начальные значения $\theta_{ij} = x_i$ для всех x_i и x_j.

В соответствии с (8.5) на шаге 3 алгоритма обновление матрицы происходит так:

$$\theta_{ij} = \begin{cases} \theta_{k,j}, \text{ если } (c_{ik} + c_{kj}) < c_{ij} \text{ в квадратных скобках} \\ \text{в выражении (8.5),} \\ \text{не изменяется, если } c_{ij} \leqslant (c_{ik} + c_{kj}). \end{cases}$$

В конце алгоритма кратчайшие пути получаются непосредственно из заключительной матрицы $\mathbf{\Theta}$. Таким образом, кратчайший путь между двумя вершинами x_i и x_j дается следующей последовательностью вершин:

$$x_i,\ x_\nu,\ \ldots,\ x_\gamma,\ x_\beta,\ x_\alpha,\ x_j,$$

где $x_\alpha = \theta_{ij}$, $x_\beta = \theta_{i\alpha}$, $x_\gamma = \theta_{i\beta}$ и т. д. до $x_i = \theta_{i_\gamma}$.

Здесь следует отметить, что если всем c_{ii} придать начальные значения ∞ (а не 0), то конечное значение величины c_{ii} будет равно весу цепи, проходящей через вершину x_i. Легко также видеть, что, исходя из структуры матрицы $\mathbf{\Theta}$, полученной в процессе той итерации, когда элемент c_{ii} становится отрицательным, легко найти цикл отрицательного веса, соответствующий этому элементу.

Это обстоятельство используется в гл. 11 при реализации основного шага в алгоритме поиска в графе потока минимальной стоимости. Оно оказывается также полезным и во многих других приложениях (об этом говорится в следующем разделе).

4. Обнаружение циклов отрицательного веса

Задача выявления циклов отрицательного веса в произвольном графе важна как сама по себе, так и в качестве основного шага в более сложных алгоритмах (см. разд. 4.1 ниже и гл. 11). В настоящем разделе эта задача обсуждается несколько более подробно, чем в предыдущем.

В разд. 3.1 отмечалось, что алгоритм Флойда нахождения кратчайших путей между всеми парами вершин может быть использован и для обнаружения в графе циклов отрицательного веса. Кроме того, в тех случаях, когда в графе имеется вершина s, из которой достижимы все остальные вершины этого графа, для нахождения циклов отрицательного веса можно также использовать алгоритм из разд. 2.2.1 (как это указывается на шаге 3(в)). Если не все вершины графа G достижимы из s (например, когда G является неориентированным графом, составленным не менее чем из двух связных компонент), то алгоритм разд. 2.2.1 завершит работу (как и должно быть) при следующем распределении пометок: у вершин из компоненты, содержащей s, пометки конечные, а пометки вершин в других компонентах равны ∞. В этом случае циклы отрицательного веса, лежащие в других компонентах, не будут обнаружены. Тем не менее во многих важных приложениях алгоритма обнаружения цикла отрицательного веса (см., например, гл. 11, разд. 5) всегда имеется в распоряжении вершина s, из которой достижимы все остальные вершины. В таких случаях алгоритм из разд. 2.2.1 при поиске циклов отрицательного веса предпочтительней с вычислительной точки зрения, чем алгоритм Флойда.

Правила остановки алгоритма из разд. 2.2.1 даются в форме, позволяющей минимизировать объем вычислений при поиске кратчайших путей из вершины s, если только в графе G отсутствуют циклы отрицательного веса. Эти правила могут быть видоизменены так, чтобы циклы отрицательного веса обнаруживались по возможности быстрее.

После каждого изменения пометки вершины x_i и нахождения подходящей вершины x_{j*}, удовлетворяющей соотношению (8.3), можно проверить, действительно ли x_i принадлежит текущему кратчайшему пути от s к x_{j*} (этот путь можно найти, используя текущие пометки θ). Если это так, то вершина x_{j*} была помечена через x_i и из неравенства $l^k(x_{j*}) + c(x_{j*}, x_i) < l^k(x_i)$ следует, что часть текущего кратчайшего пути от x_i к x_{j*} вместе с дугой (x_{j*}, x_i) должна образовывать цикл отрицательного веса и работа алгоритма может закончиться. Если же пометка вершины x_i либо не изменяется (в соответствии с (8.3)), либо изменяется, но x_i не будет принадлежать текущему кратчайшему пути от s к x_{j*},

то работа алгоритма может продолжаться до шага 3(а). Следует заметить, что описанная модификация делает излишним шаг 3(в) алгоритма из разд. 2.2.1, так как теперь цикл отрицательного веса обнаруживается сразу же после его появления, а не в конце всей процедуры.

4.1. Оптимальные циклы в графах с двойными весами [24.9]

Во многих ситуациях возникает следующая задача. Дан граф, дугам которого приписаны два числа — вес и некоторое другое число b_{ij}. Нужно найти такой цикл Ф, для которого целевая функция

$$z(\Phi)=\frac{\sum\limits_{(x_i,\ x_j)\in\Phi} c_{ij}}{\sum\limits_{(x_i,\ x_j)\in\Phi} b_{ij}}$$

минимальна (или максимальна).

Рассмотрим, например, обслуживание судном или самолетом некоторой сети маршрутов и предположим, что c_{ij} — «выгода», а b_{ij} — «время», необходимое для прохождения по дуге (x_i, x_j). Задача нахождения замкнутого обслуживаемого маршрута максимизирующего выгоду, относится к задачам описываемого типа. Более реалистичные задачи, учитывающие емкость транспортных средств и т. д., рассмотрены в работе [9].

Другими задачами, которые можно сформулировать как задачи нахождения оптимальных циклов в графах с двойными весами, являются задачи планирования параллельных вычислений [31] и управления технологическими процессами [6].

Задачу нахождения в графе с двойными весами цикла Ф, для которого отношение z^* (Ф) минимально, можно решить, используя алгоритм обнаружения в графе цикла отрицательного веса. Это делается так. Допустим, что веса c_{ij} и b_{ij} — произвольные действительные числа (положительные, отрицательные или нуль) с единственным ограничением, что для всех циклов Ф из G $\sum\limits_{(x_i,\ x_j)\in\Phi} b_{ij} > 0$. (В большинстве практических ситуаций, например, упомянутых выше, $c_{ij} \gtrless 0$, но $b_{ij} \geqslant 0$ для всех i и j.)

Возьмем некоторое пробное значение z^k целевой функции z (Ф) и рассмотрим граф G с модифицированными весами

$$c_{ij}^k = c_{ij} - z^k b_{ij}.$$

Попытка найти в G цикл отрицательного веса с матрицей $[c_{ij}^k]$ может привести к следующим трем возможностям.

А. Существует цикл Φ^- отрицательного веса, для которого

$$\sum_{(x_i,\ x_j)\in\Phi^-} c_{ij}^k < 0.$$

Б. Не существует никакого цикла с отрицательным весом и

$$\sum_{(x_i,\ x_j)\in\Phi} c_{ij}^k > 0 \text{ для всех циклов } \Phi.$$

В. Существует цикл с нулевым (но не отрицательным) весом, т. е.

$$\sum_{(x_i,\ x_j)\in\Phi^o} c_{ij}^k = 0 \text{ для некоторого цикла } \Phi^o.$$

В случае А z^* (минимальное значение z) меньше чем z^k, так как неравенство

$$\sum_{(x_i,\ x_j)\in\Phi^-} c_{ij}^k \equiv \sum_{(x_i,\ x_j)\in\Phi^-} c_{ij} - z^k \sum_{(x_i,\ x_j)\in\Phi^-} b_{ij} < 0$$

может выполняться только при

$$\frac{\sum\limits_{(x_i,\ x_j)\in\Phi^-} c_{ij}}{\sum\limits_{(x_i,\ x_j)\in\Phi^-} b_{ij}} < z_k,$$

а это, очевидно, означает, что z^* должно быть меньше чем z^k.

Аналогично в случае Б $z^* > z^k$, а в случае В $z^* = z^k$. Все сказанное приводит к следующей процедуре поиска. Начнем с некоторого значения z^1. Если это значение слишком велико (т. е. имеет место случай А), возьмем $z^2 < z^1$; если же оно слишком мало (т. е. имеет место случай Б), возьмем $z^2 > z^1$. Как только будут найдены верхняя и нижняя границы (соответственно z_u и z_l) для z^*, берем $z^k = (z_u + z_l)/2$ и заменяем в случае А z_u на z^k, а в случае Б — z_l на z^k. Так как число итераций пропорционально числу значащих разрядов требуемой точности, т. е. пропорционально $\log 1/\eta$, где η — величина погрешности, и так как каждая итерация (нахождение отрицательного цикла или вычисление полной матрицы расстояний) требует порядка n^3 операций, то решение сформулированной выше задачи с двойными весами требует $O[n^3 \log 1/\eta]$ операций.

5. Нахождение K кратчайших путей между двумя заданными вершинами

В разд. 2 были даны методы нахождения в произвольном графе кратчайшего пути от s к t. Но во многих практических приложениях требуется еще, чтобы кратчайший путь обладал некоторыми

дополнительными свойствами. Эту задачу можно, конечно, рассматривать как задачу кратчайшего пути с некоторыми дополнительными ограничениями [4] или как многоцелевую задачу, в которой учитываются не только длина пути, но и другие его свойства. Но эти усложнения будут, вообще говоря, сильно увеличивать вычислительную работу, и с практической точки зрения более просто найти K кратчайших путей от s к t и выбрать среди них тот, который обладает нужными свойствами. Хотя такой метод и не эквивалентен прямому рассмотрению дополнительных свойств пути (пример этого будет дан в разд. 7 настоящей главы), но он применим даже в тех случаях, когда эти свойства сформулированы не строго или когда они по своей природе субъективны. В этом методе предполагается, что K кратчайших путей между s и t могут быть найдены достаточно эффективно. Именно это будет предметом изучения настоящей главы.

Мы будем здесь предполагать, что рассматриваются только простые цепи. И хотя кратчайший путь необходимо должен быть простой цепью (при условии, что граф не содержит циклов отрицательного веса), но второй, третий и т. д. кратчайшие пути не обязаны быть такими даже в случае, когда все c_{ij} положительны. Задача нахождения K кратчайших путей без требования, что они являются простыми цепями, намного проще, и для ее решения Хоффман и Пэвли [20], Сакарович [32], Беллман и Калаба [3] предложили итерационные методы, аналогичные описанным выше. Однако модифицирование этих методов с целью получения простых цепей осуществить не совсем легко, а так как почти во всех практических приложениях алгоритма нахождения K кратчайших путей требуются именно простые цепи, то мы опишем метод, предложенный Йеном [35] и позволяющий находить K кратчайших *простых* цепей.

Пусть $P^k = s, x_2^k, x_3^k, \ldots, x_{q_k}^k$; t — k-й кратчайший путь от s к t, где $x_2^k, x_3^k, \ldots, x_{q_k}^k$ — соответственно 2-я, 3-я, ..., q_k-я вершины k-го кратчайшего пути. Пусть P_i^k — «отклонение от пути P^{k-1} в точке i». Под этим понимается следующее: P_i^k — кратчайший из путей, совпадающих с P_i^{k-1} от s до i-й вершины, а затем идущий к вершине, отличной от $(i+1)$-х вершин тех (ранее уже построенных) кратчайших путей P^j ($j = 1, 2, \ldots, k-1$), которые имеют те же самые начальные подпути от s к i-й вершине, что и P^{k-1}. P_i^k приходит в вершину t по кратчайшему подпути, не проходящему ни через одну из вершин $s, x_2^{k-1}, x_3^{k-1}, \ldots, x_i^{k-1}$, участвующих в формировании первой части пути P_i^k. Отсюда следует, что путь P_i^k необходимо должен быть простой цепью.

Первый подпуть $s, x_2^k, x_3^k, \ldots, x_i^k$ (совпадающий с $s, x_2^{k-1}, x_3^{k-1}, \ldots, x_i^{k-1}$) пути P_i^k называется его корнем и обозна-

чается R_i^h, а второй подпуть $x_i^h, \ldots, t$ пути P_i^h называется ответвлением и обозначается через S_i^h.

Алгоритм начинает работу с нахождения P^1 с помощью алгоритма кратчайшего пути от s к t, описанного в разд. 2. Этот путь помещается в список L_o (который должен содержать k-е кратчайшие пути). В общем случае для нахождения P^k нужно уже иметь кратчайшие пути $P^1, P^2, \ldots, P^{k-1}$. Ниже приводится описание алгоритма.

5.1. Описание алгоритма

Присвоение начальных значений

Шаг 1. Найти P^1. Положить $k = 2$. Если существует только один кратчайший путь P^1, включить его в список L_o и перейти к шагу 2. Если таких путей несколько, но меньше, чем K, включить один из них в список L_c, а остальные в список L_1. Перейти к шагу 2. Если существует K или более кратчайших путей p^1, то задача решена. Останов.

Нахождение всех отклонений

Шаг 2. Найти все отклонения P_i^k $(k-1)$-го кратчайшего пути P^{k-1} для всех $i = 1, 2, \ldots, q_{k-1}$, выполняя для каждого i шаги с 3-го по 6-й.

Шаг 3. Проверить, совпадает ли подпуть, образованный первыми i вершинами пути P^{k-1}, с подпутем, образованным первыми i вершинами любого из путей P^j $(j = 1, 2, \ldots, k-1)$. Если да, положить $c(x_i^{k-1}, x_{i+1}^j) = \infty$; в противном случае ничего не изменять. (При выполнении алгоритма вершина x_1 обозначается s). Перейти к шагу 4.

Шаг 4. Используя алгоритм кратчайшего пути, найти кратчайшие пути S_i^k от x_i^{k-1} к t, исключая из рассмотрения вершины s, x_i^{k-1}, x_3^{k-1}, $\ldots$, x_i^{k-1}. Если существует несколько кратчайших путей, взять в качестве S_i^k один из них.

Шаг 5. Построить P_i^k, соединеняя R_i^k ($\equiv s, x^{-1}, x^{-1}, \ldots, x^{-1}$ с S и поместить P в список L_1.

Шаг 6. Заменить элементы матрицы весов, измененные на шаге их первоначальными значениями и возвратиться к шагу 3.

Выбор кратчайших отклонений

Шаг 7. Найти кратчайший путь в списке L_1. Обозначить этот путь P^k и переместить его из L_1 в L_0. Если $k = K$, то алгоритм

заканчивает работу и L_0 дает требуемый список K кратчайших путей. Если $k < K$, положить $k = k + 1$ и вернуться к шагу 2.

Если в L_1 имеется более чем один кратчайший путь (скажем, h путей), то поместить в L_0 любой из них и продолжать, как и выше, до тех пор, пока увеличенное на h число путей, уже находящихся в L_1, не совпадет с K или не превысит его. Тогда алгоритм завершает работу.

Обоснование вышеприведенного алгоритма основано на очевидном факте [11], [20]: путь P^k должен быть отклонением на i-м этапе (при некотором $i \geqslant 1$) от одного из более коротких путей $P^1, P^2, \ldots, P^{k-1}$. Поэтому необходимо просто построить все кратчайшие отклонения от каждого из P^j и просмотреть их для нахождения самого короткого отклонения. Это и будет путь P^k. Следует заметить, что при k-й итерации все кратчайшие отклонения для P^j, $j = 1, 2, \ldots, k-2$, уже включены в L_1 и нужно найти только одно отклонение от P^{k-1} и пополнить им имеющийся список.

На шаге 3 мы полагаем $c(x_i^{k-1}, x_{i+1}^j) = \infty$ для тех P^j, начальные подпути которых, содержащие i вершин, совпадают с начальным i-вершинным подпутем пути P^{k-1}. Это делается для того, чтобы не получить снова P^j в качестве отклонения (в точке i) от пути P^{k-1}, что вполне могло случиться, если бы мы действовали иначе, так как при $j < k-1$ вес пути P^j не больше веса пути P^{k-1}.

Хотя на шаге 5 алгоритма каждый порожденный путь P_j^k и помещается в список L_1, совершенно очевидно, что на k-й итерации этот список не должен содержать более $K - k + 1$ кратчайших отклонений P_i^k. С вычислительной точки зрения наиболее сложным является шаг 4, где требуется $O(n^2)$ или $O(n^3)$ операций в зависимости от того, будут ли все $c_{ij} \geqslant 0$ или $c_{ij} \gtrless 0$. Так как этот шаг при k-й итерации выполняется q_k раз и так как $q_k \geqslant n$, а число итераций равно K, то рассматриваемый алгоритм при нахождении кратчайших путей требует порядка Kn^3 или Kn^4 операций. Первая величина относится к графам с неотрицательными матрицами весов, а вторая — к графам с произвольными матрицами [1]).

---

[1]) Легко видеть, что кратчайшие пути от s к t будут теми же самыми один относительно другого, если все веса c_{ij} заменить на $c'_{ij} = c_{ij} + h_i - h_j$, где величины h — произвольные числа, приписанные вершинам графа. Пользуясь этим фактом, Д. Кнут (в частном сообщении) показал, что, взяв $h_i = d(s, x_i)$, можно всегда получить $c'_{ij} \leqslant 0$. Так как расстояния $d(s, x_i)$, могут быть получены в общем случае с помощью $O(n^3)$ операций, то на самом деле (преобразуя предварительно веса) и в общем случае можно найти K кратчайших путей, используя $O(Kn^3)$ операций.

6. Кратчайший путь между двумя заданными вершинами в ориентированном ациклическом графе

Методы, развитые в предыдущих разделах этой главы, приме нимы к совершенно произвольным графам. Но на практике задачу кратчайшего пути часто требуется решать для класса ориентированных ациклических графов. Такие графы возникают в методах ПЕРТ и МКП [1]).

Допустим, что нужно реализовать некий большой проект, и этот проект состоит из большого числа этапов. Мы можем изобразить каждый этап вершиной некоторого графа и построить дугу от вершины x_i к вершине x_j, чтобы показать, что этап i должен предшествовать этапу j. Каждой дуге приписывается некоторый вес c_{ij}, равный минимальной задержке во времени между началом этапа i и началом этапа j. Пусть, например, проект представляет собой процесс строительства здания. Этап i может состоять в строительстве стен, этап j — вставка окон, этап k — прокладка проводов в стенах и т. д. Очевидно, что в этом случае нужно провести дуги от x_i к x_j и от x_i к x_k. Но минимальный срок между началом строительства стен и вставкой окон может быть отличным от срока между строительством стен и прокладкой проводов. Если, например, оконные рамы деревянные и стены должны сначала высохнуть, а для прокладки проводов это не важно, то $c_{ij} > c_{ik}$.

Совершенно очевидно, что рассматриваемый граф будет ориентированным и ациклическим. Действительно, предположение о существовании некоторого цикла, содержащего вершину x_i, приводит к (нелепому для нашей задачи) выводу о возможности повторения этапа i.

В задаче требуется найти минимальное время, необходимое для реализации проекта. Иными словами, нужно найти в графе *самый длинный путь* между вершиной s, изображающей начало, и вершиной t, изображающей завершение всех необходимых для реализации проекта работ. Самый длинный путь называется *критическим* путем, так как этапы, относящиеся к этому пути, определяют полное время реализации проекта и всякая задержка с началом выполнения любого из этих этапов приведет к задержке выполнения проекта в целом.

Сходство этой задачи с задачей о кратчайшем пути между s и t совершенно очевидно и ее можно решить, используя алгоритм

---

[1]) PERT (Project Evaluation Research Task) и CPM Critical Path Method). Для первого метода в отечественной литературе принят термин СПУ (сетевое планирование и управление). Второй метод называют методом критического пути.— *Прим. ред.*

из разд. 2.1.1 (так как все $c_{ij} \geqslant 0$), заменив все операции min на max. Но этот алгоритм был бы слишком неэкономичным, так как он не учитывает специальной структуры графа. Можно применить другой алгоритм нахождения самого длинного пути. Этот новый алгоритм описывается ниже именно в терминах самого длинного пути, так как обычно требуется найти как раз такой путь. Задача о кратчайшем пути в ориентированном ациклическом графе также может быть решена с помощью приводимого алгоритма, нужно лишь все операции max заменить на min.

6.1. Алгоритм нахождения самого длинного (критического) пути в ориентированном ациклическом графе

Пусть вершины пронумерованы так, что дуга (x_i, x_j) всегда ориентирована от вершины x_i к вершине x_j, имеющей больший номер. Для ациклического графа такая нумерация всегда возможна и производится очень легко. При этом начальная вершина получает номер 1, а конечная — номер n.

Присвоим вершине x_j пометку $l(x_j)$,— равную длине самого длинного пути от 1 до x_j, используя для этого соотношения

$$l(x_j) = \max_{x_i \in \Gamma^{-}(x_j)} [l(x_i) + c_{ij}]; \tag{8.7}$$

затем присвоим пометку вершине $(x_j + 1)$, применив снова формулу (8.7), и так далее до тех пор, пока последняя вершина n не получит пометку $l(n)$. В соотношении (8.7) $l(1)$ полагается равным нулю. Если вершина x_j помечена, то пометки $l(x_i)$ известны для всех вершин $x_i \in \Gamma^{-1}(x_j)$, так как в соответствии со способом нумерации это означает, что $x_i < x_j$ и, следовательно, вершины x_i уже помечены в процессе применения алгоритма.

Пометка $l(n)$ равна длине самого длинного пути от 1 до n. Сами дуги, образующие путь, могут быть найдены обычным способом последовательного возвращения. А именно дуга (x_i, x_j) принадлежит пути тогда и только тогда, когда $l(x_j) = l(x_i) + c_{ij}$. Начиная с вершины x_j равной n, полагаем на каждом шаге x_j равной такой вершине x_i (скажем, x_i^*), для которой выполняется это последнее равенство, и так продолжаем до тех пор, пока не будет достигнута начальная вершина (т. е. пока не будет $x_i^* = 1$).

Совершенно очевидно, что пометка $l(x_j)$ вершины x_j дает длиннейший путь от 1 до x_j, т. е. самое раннее возможное начало выполнения этапа, изображаемого вершиной x_j. Таким образом, если (x_i, x_j) — дуга в графе, то $l(x_j) - l(x_i) - c_{ij}$ (неотрицательная величина, как это видно из (8.7)) дает самое большое возможное время задержки начала этапа i, не оказывающее влияния на время начала этапа j. Из вышесказанного видно, что если i

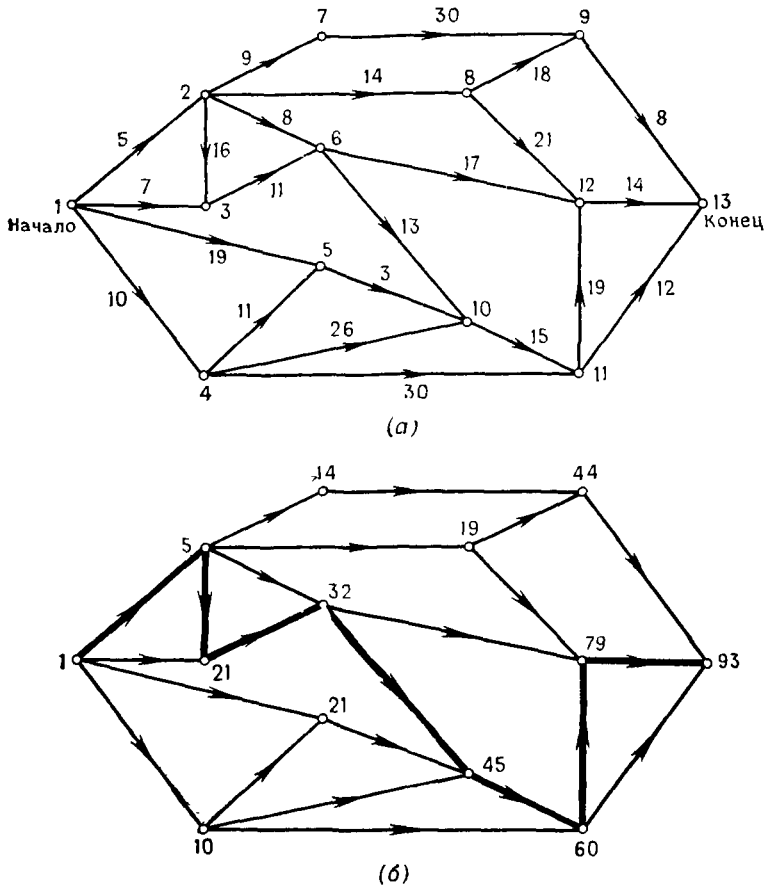


Рис. 8.5. (а) Граф из примера 6.1.1. (б) Окончательные пометки вершин и критический путь.

и j — два последовательные этапа в самом длинном пути, то $l(x_j) - l(x_i) - c_{ij} = 0$.

6.1.1. Пример. Рассмотрим диаграмму ПЕРТ, изображенную на рис. 8.5 (а), где числа, стоящие у дуг, указывают продолжительность задержек в днях. Каков наименьший возможный срок c_{ij} выполнения этого проекта? (Предполагается, что вершина 13 является конечной и ей соответствует нулевая продолжительность.) Вершины графа уже пронумерованы так, что дуга (x_i, x_j) существует лишь при $x_j > x_i$. Применение выражения (8.7) к графу, изображенному на рис. 8.5 (а), приводит к расстановке

пометок, показанной на рис. 8.5 (б); критический путь изображен жирной линией. Длина этого пути и, следовательно, наименьший возможный срок выполнения проекта равен 93 дням.

Если один из этапов, принадлежащих критическому пути, задерживается на D дней, то и весь проект задержится на тот же срок. С другой стороны, задержка, скажем, этапа 8 (не принадлежащего критическому пути) возможна на 79—19—21 = 39 дней, причем это не отразится на общем сроке реализации проекта. Лишь задержка на срок $D > 39$ дней повлияет на пометку вершины 12 (принадлежащую критическому пути).

Метод ПЕРТ описан в этой главе в терминах самого длинного пути в графе, вершины которого отвечают этапам выполнения проекта, а дуги — отношениям предшествования, причем вес c_{ij} дуги (x_i, x_j) равен интервалу времени между началом этапа i и j. На практике диаграммы ПЕРТ истолковываются несколько иначе, а именно дуги представляют этапы, а вершины изображают абстрактные «события», указывающие начала или окончания этапов. Хотя такое представление само по себе является неполным (так как при этом отношения предшествования точно не описываются), указанный недостаток можно устранить, добавляя фиктивные этапы для точного описания отношений предшествования [34]. С другой стороны, такое представление имеет важное преимущество перед первоначальным, поскольку приводит на практике к более простым диаграммам. Это происходит в силу того, что временная задержка c_{ij} между началами этапов i и j является, вообще говоря, постоянной для данного i и совершенно не зависит от следующего этапа. Редкие исключения всегда можно учесть, вводя некоторые фиктивные этапы. В таких случаях подобное представление приводит на практике к более простым графам (даже с учетом фиктивных вершин), в то время как представление, рассмотренное ранее, остается неизменным. Другой особенностью практического использования метода ПЕРТ является то, что интервалы времени (т. е. временные задержки) между этапами считаются случайными величинами, а не детерминированными, как считалось здесь.

Задачи планирования, решаемые с помощью метода ПЕРТ, не учитывают наличие ресурсов и не рассматривают потребности этапов в ресурсах. Вместо этого производится (после нахождения критического пути) такое распределение ресурсов по этапам проекта, при котором задерживается выполнение этапов, не принадлежащих критическому пути. Если же ограничения на ресурсы учитывать на стадии планирования, то задача из очень легкой превращается в очень трудную [1, 30, 37] и решить ее удается лишь для размерностей на 3—4 порядка меньших, чем в случае соответствующей задачи без ограничений.

7. Задачи, близкие к задаче о кратчайшем пути

В алгоритме Флойда нахождения кратчайшего пути между всеми парами вершин (см. разд. 3 настоящей главы) была использована трехместная операция (8.5). Применение этой операции n раз к матрице весов $[c_{ij}]$ дает «заключительную» матрицу, элементы которой равны длинам кратчайших путей. Эта операция является частным случаем более общей трехместной операции

$$z_{ij} = \mathrm{OPT}\,[z_{ij},\ z_{ik} \otimes z_{kj}], \tag{8.8}$$

где z — функция пути, подлежащая оптимизации, а $\otimes$ — некоторая общая операция. Операция $\otimes$ налагает единственное ограничение в случае ее применения к различным задачам нахождения кратчайших путей в графах. А именно если путь от x_i к x_j проходит через промежуточную вершину x_k и z — некоторая характеристика пути, на которой базируется оптимизация, то для $\otimes$ должно выполняться условие

$$z_{ij} = z_{ik} \otimes z_{kj}. \tag{8.9}$$

Ниже приводится несколько примеров использования этой операции.

7.1. Наиболее надежный путь

В задаче о кратчайшем пути в качестве «длины» пути бралась сумма весов дуг, образующих этот путь. Рассмотрим теперь случай, когда «вес» дуги представляет ее надежность. Надежность пути от s к t, составленного из дуг, взятых из множества P, дается формулой

$$\rho\,(P) = \prod_{(x_i,\ x_j) \in P} \rho_{ij}, \tag{8.10}$$

где ρ_{ij} — надежность дуги $(x_i,\ x_j)$, т. е. вероятность ее существования в графе (или — в случае физических систем — вероятность того, что она находится в работоспособном состоянии).

Задачу нахождения наиболее надежного пути от s к t можно свести к задаче о кратчайшем пути от s к t, взяв в качестве «веса» c_{ij} дуги $(x_i,\ x_j)$ величину $c_{ij} = -\log \rho_{ij}$. Прологарифмировав обе части соотношения (8.10), получим

$$\log \rho\,(P) = \sum_{(x_i,\ x_j) \in P} \log \rho_{ij} = -\sum_{(x_i,\ x_j) \in P} c_{ij}.$$

Отсюда видно, что кратчайший путь от s к t с матрицей весов $[c_{ij}]$ будет в то же время и наиболее надежным путем с матрицей $[\rho_{ij}]$, а надежность этого пути равна антилогарифму его длины.

При другой формулировке задачи о наиболее надежном пути возможно непосредственное использование трехместной операции; а именно операция вводится с помощью соотношения

$$\rho_{ij} = \max [\rho_{ij},\ \rho_{ik} \cdot \rho_{kj}].$$

В качестве начальной $[\rho_{ij}]$-матрицы берется матрица надежностей дуг, причем нулевые элементы указывают на отсутствие соответствующих дуг. Такая формулировка позволяет, очевидно, найти наиболее надежные пути между всеми парами вершин.

7.2. Путь с наибольшей пропускной способностью

В этой задаче каждая дуга (x_i, x_j) графа имеет пропускную способность q_{ij} и требуется найти путь от s к t с наибольшей пропускной способностью. Пропускная способность пути P определяется, конечно, дугой из P с наименьшей пропускной способностью, т. е.

$$Q(P) = \min_{(x_i,\ x_j) \in P} [q_{ij}]. \tag{8.11}$$

Теорема 1 [1]). *Пропускная способность пути с наибольшей пропускной способностью от s к t равна*

$$\min_K \{ \max_{(x_i,\ x_j) \in K} [q_{ij}] \},$$

где K — любой $(s\text{-}t)$-разрез (в множестве дуг).

Доказательство. Пусть $\hat{Q}$ — пропускная способность пути с наибольшей пропускной способностью. Так как каждый путь от s к t должен содержать по крайней мере одну дугу из каждого $(s\text{-}t)$-разреза, то для каждого разреза K будет выполняться неравенство

$$\max_{(x_i,\ x_j) \in K'} [q_{ij}] \leqslant \hat{Q}. \tag{8.12}$$

Поскольку в каждом $(s\text{-}t)$-пути должна присутствовать по крайней мере одна дуга с пропускной способностью, не превосходящей $\hat{Q}$, и поскольку, взяв по одной такой дуге из каждого $(s\text{-}t)$-пути, мы получаем некоторый $(s\text{-}t)$-разрез (по определению), то существует хотя бы один разрез, скажем K', для которого

$$\max_{(x_i,\ x_j) \in K'} [q_{ij}] \leqslant \hat{Q}. \tag{8.13}$$

Таким образом, разрез $\hat{K}$, доставляющий минимум выражению

$$\min_K \{ \max_{(x_i,\ x_j) \in K} [q_{ij}] \}, \tag{8.14}$$

[1]) Обратите внимание на сходство теоремы 1 с теоремой о максимальном потоке и минимальном разрезе из гл. 2; на самом деле теорема 1 является минимаксным вариантом последней.

должен одновременно удовлетворять неравенствам (8.12) и (8.13), т. е. для него должно выполняться равенство. Отсюда следует справедливость теоремы.

Очевидный алгоритм нахождения пути с наибольшей пропускной способностью, основанный на теореме 1, состоит в следующем.

Шаг 1. Начать с $(s\text{-}t)$-разреза $\bar{K} \equiv (\{s\}, X - \{s\})$ и найти наибольшую пропускную способность $\bar{Q}$ дуг из $\bar{K}$.

Шаг 2. Построить остовный подграф $G' = (X, A')$, где $A' = \{(x_i, x_j) \mid q_{ij} \geqslant \bar{Q}\}$.

Шаг 3. Найти множество $R'(s)$ вершин, достижимых из s в графе A'. (Построение множества $R'(s)$ описано в гл. 2.)

Шаг 4. Если $t \in R'(s)$, то $\hat{Q} = \bar{Q}$ и любой $(s\text{-}t)$-путь в остовном подграфе G' будет иметь наибольшую пропускную способность, равную $\hat{Q}$. Если $t \notin R'(s)$, перейти к шагу 5.

Шаг 5. Взять в качестве $\bar{K}$ разрез $(R'(s), X - R'(s))$ и найти наибольшую пропускную способность $\bar{Q}$ дуг в этом новом разрезе. (Текущее значение величины $\bar{Q}$ меньше, чем предыдущее, в силу определения множества $R'(s)$ на шаге 3 и множества дуг A' на шаге 2.) Возвратиться к шагу 2.

Для неориентированных графов G Франк и Фриш [16] предложили еще более простую процедуру нахождения пути с наибольшей пропускной способностью, состоящую в следующем.

Пусть K_1 — некоторый $(s\text{-}t)$-разрез, а $\bar{Q}$ — наибольшая пропускная способность ребер из K_1. Если теперь «закоротить» любое ребро (x_i, x_j) с пропускной способностью $q_{ij} \geqslant \bar{Q}$, т. е. вершины x_i и x_j заменить одной вершиной x, положив

$$\Gamma(x) = \Gamma(x_i) \cup \Gamma(x_j), \quad \Gamma^{-1}(x) = \Gamma^{-1}(x_i) \cup \Gamma^{-1}(x_j),$$

и ребро (x_i, x_j) удалить, то получившийся граф G_1 будет иметь тот же самый путь с наибольшей пропускной способностью, что и первоначальный граф G. Справедливость этого утверждения вытекает из того факта, что $\bar{Q}$ является верхней границей для $\hat{Q}$ (в соответствии с (8.12)), и, следовательно, ребра с $q_{ij} \geqslant \bar{Q}$ не могут повлиять на оптимальное решение, т. е. не могут войти в разрез $\hat{K}$. Поскольку «закорачивание» ребра (x_i, x_j) может повлиять только на разрезы, содержащие (x_i, x_j) (эти разрезы удаляются), то разрез $\hat{K}$ графа G (или все разрезы, доставляющие минимум выражению (8.14), если их несколько) будет также и разрезом преобразованного графа G_1.

Процедуру, примененную к первоначальному графу G, можно повторить и для графа G_1, выбирая другой $(s\text{-}t)$-разрез K_2, зако-

рачивая все ребра из G_1, пропускная способность которых не меньше наибольшей пропускной способности любого ребра из K_2. Это дает граф G_2 и т. д. Процесс закончится, когда будут закорочены s и t. Теперь каждый $(s\text{-}t)$-путь в графе $\hat{G}$, образованный вершинами из G и теми ребрами, которые оказались закороченными, будет иметь максимально возможную пропускную способность $\hat{Q}$. Ограничение на применимость данного алгоритма только к неориентированным графам вызвано именно процессом «закорачивания», так как при этом неявно предполагается, что путь с пропускной способностью, не меньшей, чем $\bar{Q}$, существует между любыми вершинами (двумя или большим числом), заменяемыми в описанном процессе единственной вершиной, но такое предположение может не выполняться, если закорачиваемые ребра имеют ориентацию.

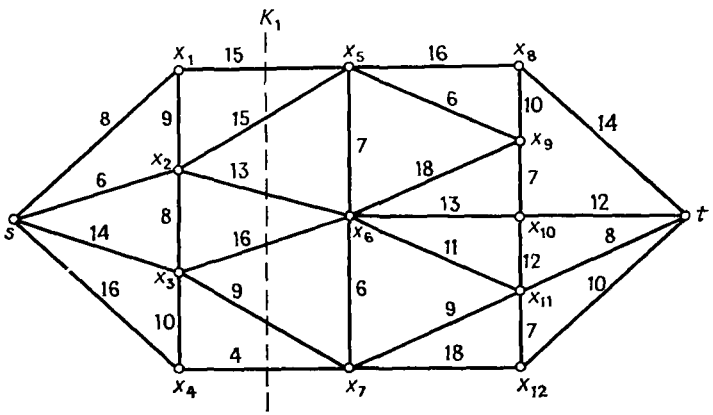


Рис. 8.6а. Граф из примера 7.2.1.

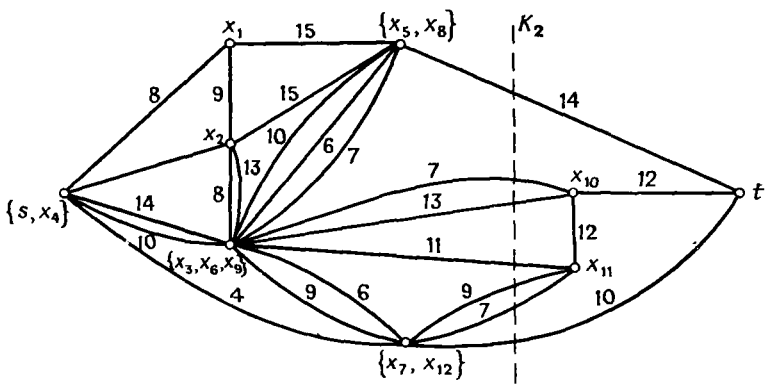


Рис. 8.6б. Граф после первого стягивания.

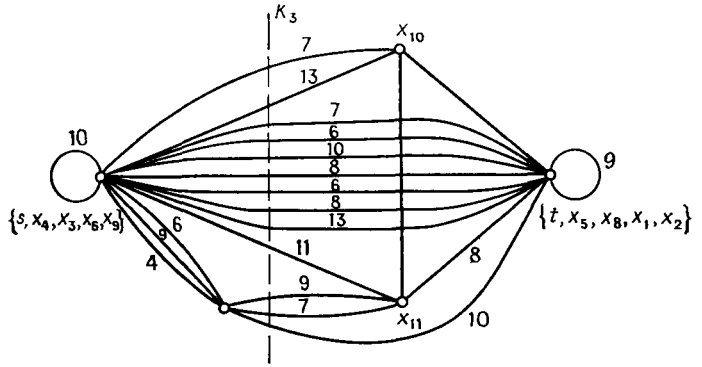


Рис. 8.6в. Граф после второго стягивания.

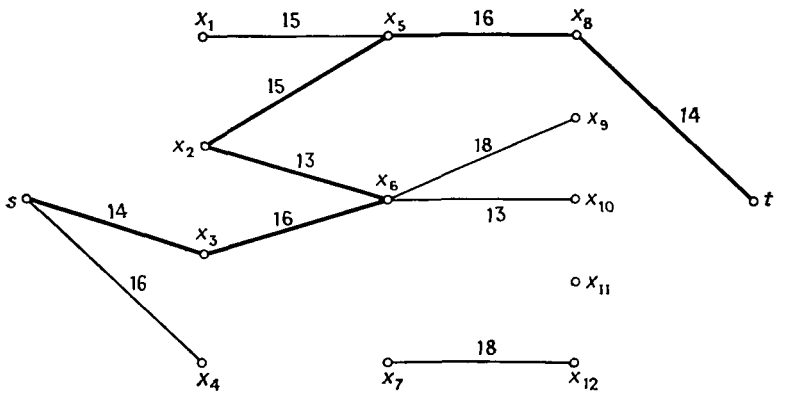


Рис. 8.6г. Граф $\hat{G}$ и путь с наибольшей пропускной способностью.

7.2.1. Пример. Задача состоит в нахождении $(s\text{-}t)$-пути (s) наибольшей пропускной способности в неориентированном графе, изображенном на рис. 8.6а, где пропускные способности ребер указаны числами, стоящими у этих ребер.

Выбрав произвольно $(s\text{-}t)$-разрез K_1, показанный на рис. 8.6а пунктиром, найдем, что ребром с максимальной пропускной способностью в K будет (x_3, x_6) с $q_{3,6} = 16$. Закорачивая все ребра с пропускной способностью $\geqslant 16$, получим граф G_1, изображенный на рис. 8.6б. Выберем теперь $(s\text{-}t)$-разрез K_2 графа G_1 так, как показано пунктиром на рис. 8.6б. Наибольшая пропускная способность ребер в K_2 равна 14. Поэтому закорачиваем все ребра с пропускной способностью $\geqslant 14$ и получаем граф G_2, изображенный на рис. 8.6в. Беря в качестве следующего $(s\text{-}t)$-разреза K_3 — он показан пунктиром на рис. 8.6в,— получаем $\hat{Q} = 13$, а в по-

строенном графе (полученном из G_2 после закорачивания всех ребер с пропускной способностью $\geqslant 13$) вершины s и t будут закорочены.

Окончательным графом $\hat{G}$ будет остовный подграф графа G, содержащий лишь те ребра, которые закорачивались во время описанной процедуры (т. е. только ребра (x_i, x_j) с $q_{ij} \geqslant 13$). Этот последний граф изображен на рис. 8.6г и видно, что существует только один (s-t)-путь с наибольшей пропускной способностью. Этот путь показан жирными линиями, причем критическим ребром этого пути будет (x_2, x_6) с пропускной способностью 13.

Если требуется найти пути с наибольшей пропускной способностью между каждой парой вершин графа, то трехместную операцию из выражения (8.8) можно использовать в форме

$$q_{ij} = \max [q_{ij}, \min \{q_{ik}, q_{kj}\}], \tag{8.15}$$

поскольку так введенная операция удовлетворяет требованиям, налагаемым соотношением (8.9). Начальной матрицей $[q_{ij}]$ является исходная матрица пропускных способностей дуг (с нулевыми элементами на местах отсутствующих дуг); конечная матрица $[q_{ij}]$ дает пропускные способности путей между всеми парами вершин.

7.3. Путь с наибольшей приведенной пропускной способностью

Рассмотрим граф G, в котором каждой дуге (x_i, x_j) приписаны два числа p_{ij} и q_{ij}, представляющие соответственно надежность и пропускную способность дуги. Задача нахождения пути от s к t с наибольшей приведенной пропускной способностью является комбинацией двух последних задач о путях, обсуждавшихся выше в разд. 7.1 и 7.2. Если приведенную пропускную способность пути P обозначить через $e(P)$, то задача состоит в нахождении пути, минимизирующего выражение

$$e(P) = \prod_{(x_i, x_j) \in P} p_{ij} \cdot \{\min_{(x_i, x_j) \in P} [q_{ij}]\}. \tag{8.16}$$

В этом случае нельзя воспользоваться трехместной операцией, так как не удается найти оператора $\otimes$, который (для пути от x_a к x_b через вершину x_c) удовлетворяет условию $e_{ab} = e_{ac} \otimes e_{cb}$.

Несуществование такого оператора можно обосновать следующим образом:

$$e_{ab} = \prod_{(x_i, x_j) \in a \to b} p_{ij} \cdot \min_{(x_i, x_j) \in a \to b} [q_{ij}] =$$

$$= \prod_{(x_i, x_j) \in a \to c} p_{ij} \cdot \prod_{(x_i, x_j) \in c \to b} p_{ij} \cdot \min \{\min_{(x_i, x_j) \in a \to c} [q_{ij}], \min_{(x_i, x_j) \in c \to b} [q_{ij}]\},$$

где запись $a \to b$ и т. д. применяется как для пути от x_a к x_b, так и для множества дуг этого пути.

Вводя обозначения

$$\bar{\rho}_{ac} = \prod_{(x_i,\, x_j) \in a \to c} \rho_{ij}, \quad \bar{\rho}_{cb} = \prod_{(x_i,\, x_j) \in c \to b} \rho_{ij},$$

$$\bar{q}_{ac} \min_{(x_i,\, x_j) \in a \to c} [q_{ij}], \quad \bar{q}_{cb} = \min_{(x_i,\, x_j) \in c \to b} [q_{ij}],$$

можем записать

$$e_{ac} = \bar{\rho}_{ac} \cdot \bar{q}_{ac}, \quad e_{cb} = \bar{\rho}_{cb} \cdot \bar{q}_{cb}.$$

Следовательно,

$$e_{ab} = \min [\bar{\rho}_{ac} e_{cb}, \bar{\rho}_{cb} e_{ac}]. \tag{8.17}$$

Из этого выражения видно, что так как надежности $\bar{\rho}_{ac}$ и $\bar{\rho}_{cb}$ не могут быть выражены «на языке» величин e_{ac}, e_{cb}, ρ_{ij} и q_{ij}, то невозможно найти оператор, удовлетворяющий общим условиям

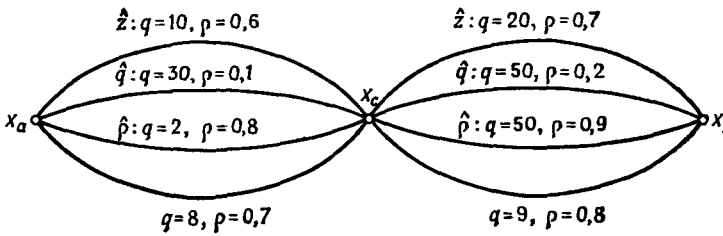


Рис. 8.7.

(8.9) и, следовательно, невозможно непосредственно использовать трехместную операцию.

На самом деле нетрудно показать, что оптимальный путь от x_a к x_b через x_c не зависит от:

(i) оптимальных путей от x_a к x_c и от x_c к x_b;

(ii) путей наибольшей пропускной способности от x_a к x_c и от x_c к x_b;

(iii) путей наибольшей надежности от x_a к x_c и от x_c к x_b или любой их комбинации.

В этом можно убедиться, рассматривая рис. 8.7, на котором изображены пути всех трех вышеуказанных типов от x_a к x_c и от x_c к x_b, причем оптимальное значение величины e, которое можно получить с помощью некоторой комбинации этих путей, равно 4,20. На рисунке изображены, кроме того, такие два пути от x_a к x_c и от x_c к x_b, ни один из которых не является оптимальным в любом из трех вышеприведенных смыслах, между их соответствующими конечными вершинами. Однако эти два пути образуют оптимальный путь между вершинами x_a и x_b, которому отвечает значение величины e, равное 4,48.

7.3.1. Метод нахождения пути наибольшей приведенной пропускной способности

Здесь мы дадим итерационный метод нахождения пути с наибольшей приведенной пропускной способностью от некоторой указанной вершины s в другую вершину t графа $G = (X, A)$. Этот метод состоит в постепенном исключении тех дуг графа, которые не могут принадлежать оптимальному пути. Исключение осуществляется до тех пор, пока граф не станет несвязным.

Сначала находим путь $P_{\hat{\rho}}$ между s и t, имеющий наибольшую надежность. Пусть $Q_{\hat{\rho}}$ — пропускная способность этого пути, т. е.

$$Q_{\hat{\rho}} = \min_{(x_i,\, x_j) \in P_{\hat{\rho}}} [q_{ij}].$$

При этом через $P_{\hat{\rho}}$ будет обозначаться также и множество дуг, образующих рассматриваемый путь. Если из A удалить множество дуг $A_0 \equiv \{(x_i, x_j) \mid (x_i, x_j) \in A,\ q_{ij} \leqslant Q_{\hat{\rho}}\}$ и образовать множество $A' = A - 0$, то либо граф $G' = (X, A')$, являющийся остовным подграфом графа G, содержит оптимальный путь из G, либо $P_{\hat{\rho}}$ будет оптимальным путем. Это можно обосновать так.

Оптимальный путь в G должен иметь по определению надежность, не большую, чем надежность пути $P_{\hat{\rho}}$. Следовательно, значение приведенной пропускной способности должно быть не меньше, чем значение этой величины для $P_{\hat{\rho}}$. Если $P_{\hat{\rho}}$ — не оптимальный путь, то пропускная способность оптимального пути в G больше, чем $Q_{\hat{\rho}}$, а значит, ему не принадлежит никакая дуга из A_0.

Далее уже в графе G' ищется путь $P'_{\hat{\rho}}$ с наибольшей надежностью. Пропускная способность $Q'_{\hat{\rho}}$ этого пути больше, чем $Q_{\hat{\rho}}$, но его надежность не превосходит надежности пути $P_{\hat{\rho}}$. Если значение приведенной пропускной способности пути $P'_{\hat{\rho}}$ больше, чем значение этой величины для $P_{\hat{\rho}}$, то $P'_{\hat{\rho}}$ берется как лучший путь и хранится до тех пор, пока не будет найден путь с большей приведенной пропускной способностью. Затем из A удаляется соответствующее множество дуг

$$A_1 = \{(x_i, x_j) \mid (x_i, x_j) \in A',\ q_{ij} \leqslant Q'_{\hat{\rho}}\}.$$

Получается множество $A'' = A' - A_1$ и остовный подграф $G'' = (X, A'')$. По тем же самым причинам, что и выше, либо граф G'' содержит оптимальный путь, либо оптимальным будет наилучший из найденных до данного момента путь. Этот процесс продолжается до тех пор, пока не получится остовный подграф G^l, удовлетворяющий одному из следующих условий: либо он является несвязным,

причем в нем нет пути между s и t, либо приведенная пропускная способность наилучшего из найденных ранее путей будет больше, чем произведение надежности пути $P^l_{\hat{\rho}}$ на пропускную способность такого пути в графе G, который обладает наибольшей пропускной способностью (этот путь можно найти с помощью методов, описанных выше); тогда, очевидно, никакой остовный подграф графа G^l не приводит к лучшему решению.

Оптимальным путем будет текущий наилучший путь, полученный в конце описанной процедуры.

На эффективность вышеприведенного метода влияют два фактора. Первый — «скорость» удаления дуг из графа В худшем

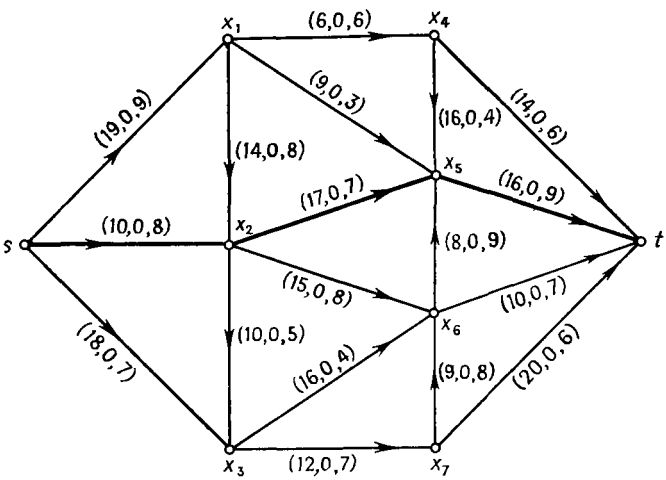


Рис. 8.8. Граф из примера 7.3.2.

случае, когда на каждом этапе наиболее надежный путь использует дугу с наименьшей пропускной способностью, понадобится $m - k$ этапов (для нахождения пути с наилучшей надежностью), где m — число дуг в G и k — число дуг в оптимальном пути Самым хорошим случаем будет тот, когда уже первый остовный подграф G' является несвязным и вершины s и t находятся в разных компонентах. Тогда потребуется только один этап

Второй фактор связан с тем методом, который используется для «перевычисления» (на каждом этапе) пути с наибольшей надежностью в остовном подграфе. При этом следует заметить, что хотя удаление дуг может привести, вообще говоря, к элиминации нескольких дуг из s-базы предшествующего графа (дающего наиболее надежные пути от s ко всем другим вершинам), но та

часть s-базы, которая содержит s вместе с пометками вершин, остается неизменной и ее не нужно находить заново.

7.3.2. Пример. Найти путь от s к t с наибольшей приведенной пропускной способностью в графе G, изображенном на рис. 8.8, где каждая дуга имеет пометку (a, b), причем a является пропускной способностью дуги, а b — ее надежностью.

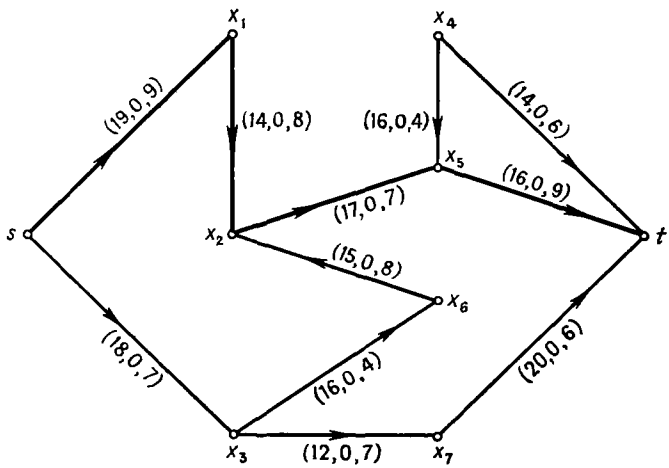


Рис. 8.9а. Граф G' из примера 7.3.2.

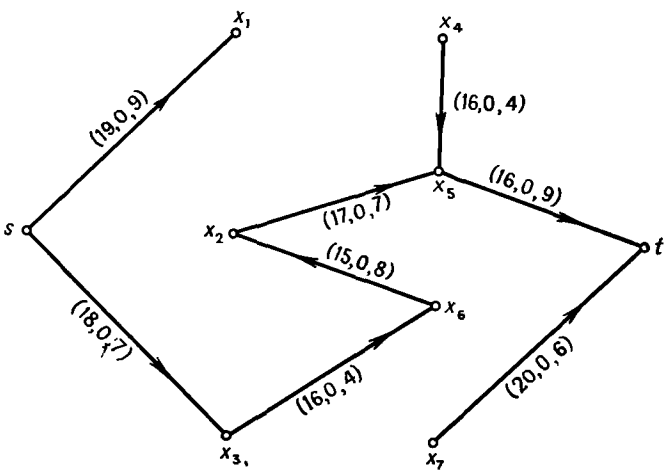


Рис. 8.9б. Граф G'' из примера 7.3.2.

Путь $P_{\hat{\rho}}$ с наибольшей надежностью в графе G показан на на рис. 8.8 жирными линиями. Здесь $\hat{\rho} = 0{,}504$, $Q_{\hat{\rho}} = 10$ и, следовательно, приведенная пропускная способность этого пути равна $e = 5{,}04$. Удаляя из G все дуги с пропускными способностями $\leqslant 10$, получим граф G', изображенный на рис. 8.9а. Путь $P'_{\hat{\rho}}$ с наибольшей надежностью в графе G' показан на рисунке жирными линиями. Для этого пути $\hat{\rho} = 0{,}454$, $Q'_{\hat{\rho}} = 14$ и, значит, приведенная пропускная способность пути $P'_{\hat{\rho}}$ равна $e = 14 \times \times 0{,}454 = 6{,}36$, что больше полученной ранее величины (5,04) и, следовательно, заменяет ее. Удаляя из G' все дуги с пропускной способностью $\leqslant 14$, получаем граф G'', изображенный на рис. 8.9б. В этом графе существует только один путь между s и t; $\hat{\rho} = 0{,}141$, $Q''_{\hat{\rho}} = 15$ и, следовательно, приведенная пропускная способность равна 2,12, что хуже, чем лучшее предыдущее значение этой величины. Удаляя все дуги с пропускной способностью $\leqslant 15$ из G'', мы разъединим вершины s и t, так что лучший предыдущий ответ, т. е. путь (s, 1, 2, 5, t), с приведенной пропускной способностью 6,36 будет оптимальным ответом.

8. Задачи

1. Найти кратчайшие пути от вершины 1 ко всем другим вершинам графа на рис. 8.10. Изобразить также 1-базу.

2. Используя метод из разд. 2.2, найти кратчайшие пути от вершины 1 ко всем другим вершинам графа, изображенного на рис. 8.11.

3. Решить задачу 2, изменив вес ребра (4,5) с 8 на —8. Выявить все отрицательные циклы.

4. Граф на рис. 8.12 представляет сеть допустимых маршрутов для некоторого судна. Каждая дуга имеет пометку (a, b), причем a равно выгоде, получаемой при обслуживании этого маршрута, а b — времени обслуживания маршрута. Найти наиболее выгодный (в терминах скорости оборота капитала) маршрут судна.

5. Для графа, изображенного на рис. 8.10, найти четыре кратчайшие простые цепи от вершины 1 к вершине 10. Найти кратчайшую простую цепь из 5 дуг от вершины 1 к вершине 10.

6. На рис. 8.13 изображена диаграмма ПЕРТ. Продолжительность операции i дается числом, стоящим у дуги, исходящей из соответствующей вершины i. Найти критический путь и такую максимально возможную задержку начала выполнения операции 5, которая не приведет к задержке реализации всего проекта.

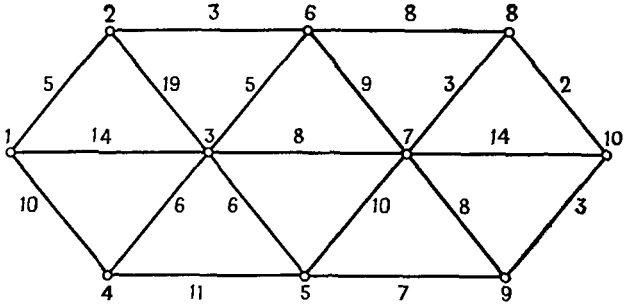


Рис. 8.10.

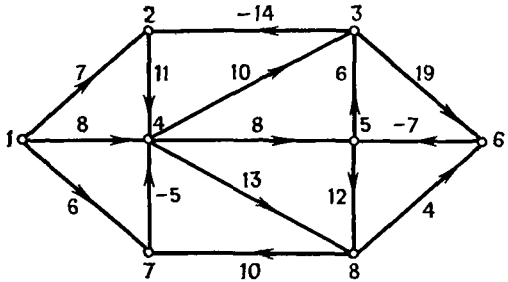


Рис. 8.11.

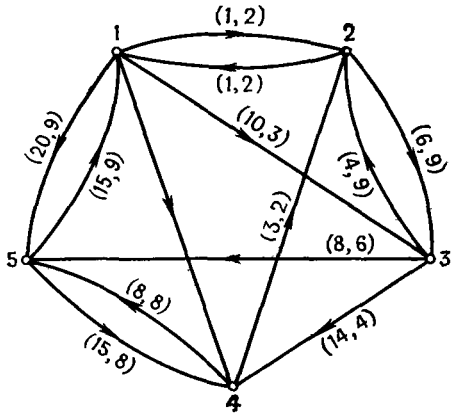


Рис. 8.12.

7. В графе, изображенном на рис. 8.14, каждое ребро имеет пометку (a, b), где a — пропускная способность, а b — надежность ребра. Найти:

(а) путь с наибольшей пропускной способностью от вершины 1 к вершине 10;

(б) путь с наибольшей надежностью от вершины 1 к вершине 10;

(в) путь с наибольшей приведенной пропускной способностью от вершины 1 к вершине 10

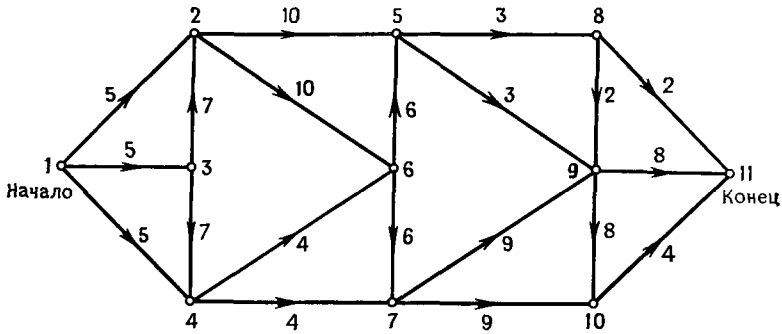


Рис. 8.13.

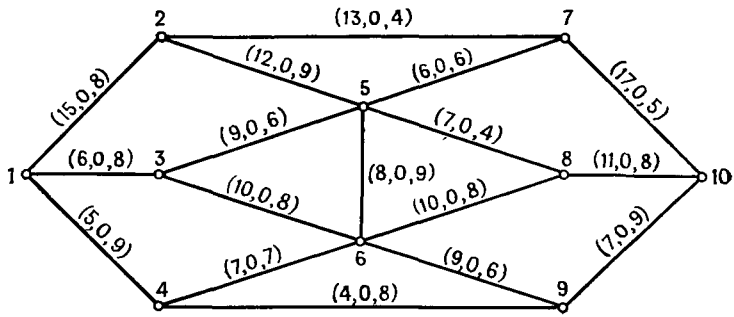


Рис. 8.14.

8. Для того случая, когда веса всех дуг неотрицательны, модифицировать алгоритм Дейкстры так, чтобы расстановка пометок происходила одновременно и от вершины s, и от вершины t. Уделить особое внимание правилу окончания процедуры (см. [29], [28]).

9. Доказать, что ответ, полученный по алгоритму Флойда, описанному в разд. 3.1, является оптимальным, и показать, что объем вычислений растет как $O(n^8)$ (см. [71]).

10. Пусть T — ориентированный остов графа $G = (X, A)$ и корнем остова T является вершина s. Пусть $l(x_i)$ — расстояние от вершины s до вершины x_i, измеренное вдоль пути в этом дереве. Показать, что T является s-базой тогда и только тогда, когда для каждой дуги $(x_i, x_j) \in A$ выполняется условие $l(x_j) \leqslant l(x_i) + c_{ij}$.

11. Теорема из задачи 10 является основой следующей итерационной процедуры нахождения кратчайшего пути от s ко всем другим вершинам графа (здесь мы предполагаем, что все вершины $x_i \in X$ достижимы из s).

Шаг 1. Взять в качестве T любой ориентированный остов графа G и для каждой вершины x_i найти пометку $l(x_i)$, равную расстоянию между этой вершиной и корнем s вдоль пути в дереве T.

Шаг 2. Если каждая дуга $(x_i, x_j) \in A$ удовлетворяет условию

$$l(x_j) \leqslant l(x_i) + c_{ij},$$

то остановиться. Текущий остов T будет s-базой графа G. В противном случае перейти к шагу 3.

Шаг 3. Если для некоторой дуги (x_i', x_j')

$$l(x_j') > l(x_i') + c_{i'j'},$$

то добавить к дереву дугу (x_i', x_j') и удалить из него ту дугу (x, x_j'), конечной вершиной которой является x_j', после чего возникает новое дерево T.

Шаг 4. Обновить пометки $l(x_k)$ у всех вершин x_k, достижимых из x_j' вдоль путей дерева T. Возвратиться к шагу 2.

Показать, что вышеописанный алгоритм является конечным, и показать, что его сложность для полного n-вершинного графа растет как $O(n^3)$. Налагает ли алгоритм какие-либо ограничения на веса c_{ij}? Используя этот алгоритм, проверить решение задачи 2 (см. [14]).

9. Список литературы

1. Balas E. (1969), Project scheduling with resources constraints, IBM New York Scientific Center, Report No. 320—2960.
2. Bellman R. (1958), On a routing problem, *Quart. of Applied Mathematics*, **16**, p. 87.
3. Bellman R., Kalaba R. (1960), On k^{th}-best policies, *J. of SIAM*, 8, p. 582.
4. Berry R. C. (1971), A constrained shortest path algorithm, Paper presented at the 39th National ORSA Meeting, Dallas, Texas.
5. Brucker P. (1972), All shortest distances in networks with a large number of strong components, Presented at the 41st National Meeting Research Soc. of America, New Orleans.

6. Cunninghame-Green R. A. (1962), Describing industrial processes with interference and approximating their steady-state behaviour, *Opl. Res. Quart.*, **13**, p. 95.
7. Dantzig G. B. (1966), All shortest routes in a graph, Int. Symp. on Theory of Graphs, Dunod, Paris, p. 91.
8. Dantzig G. B., Blattner W. O., Rao M. R. (1966), All shortest routes from a fixed origin in a graph, Int. Symp. on Theory of Graphs, Dunod, Paris, p. 85.
9. Dantzig G. B., Blattner W. O., Rao M. R. (1966), Finding a cycle in a graph with minimum cost to time ratio with applications to a ship routing problem, Int. Symp. on Theory of Graphs, Dunod, Paris, p. 77.
10. Dijkstra E. W. (1959), A note on two problems in connection with graphs, *Numerische Mathematik*, **1**, p. 269.
11. Dreyfus S. E. (1969), An appraisal of some shortest path algorithms, *Ops. Res.*, **17**, p. 395.
12. Ermolev Yu. M. (1966), Shortest admissible paths I, *Kibernetika*, **2**, p. 88.
13. Floyd R. W. (1962), Algorithm 97 — Shortest path, *Comm. of ACM*, **5**, p. 345.
14. Ford L. R. Jr. (1946), Network flow theory, Rand Corporation Report P-923.
15. Fox B. (1969), Finding a minimal cost to time ratio circuit, *Ops. Res.*, **17**, p. 546.
16. Frank H., Frisch I. T. (1971), Communication, Transmission and Transportation Networks, Addison-Wesley, Reading, Massachusetts.
17. Gill A., Traiger T. (1968), Computation of optimal paths in finite graphs, Proc. 2nd Int. Conf. on Comp. Methods in Optimization Problems, San Remo, Italy.
18. Hitchener L. E. (1968), A comparative investigation of the computational efficiency of shortest path algorithms, Operations Research Centre, University of California, Berkeley, Report ORC 68-17.
19. Hoffman A. J., Winograd S. (1971), On finding all shortest distance in a directed network, IBM, Thomas J. Watson Research Center, Report RC 3613.
20. Hoffman W., Pavley R. (1959), A method for the solution of the N^{th} best path problem, *J. of ACM*, **6**, p. 506.
21. Hu T. C. (1969), Integer programming and network flows, Addison-Wesley, Reading, Massachusetts.
22. Johnson E. L. (1972), On shortest paths and sorting, Proc. of Annual Conf. of ACM, Boston, p. 510.
23. Joksch H. C. (1966), The shortest route problem with constraints, *J. of Math. Anal. and Appl.*, **14**, p. 191.
24. Lawler E. L. (1966), Optimal cycles in doubly weighted directed linear graphs, Int. Symp. on Theory of Graphs, Dunod, Paris, p. 209.
25. Minieka E. (1971), All k-shortest paths in a graph, Internal Report, Department of Statistics, Trinity College, Dublin.
26. Moore E. F. (1957), The shortest path through a maze, Proc. Int. Symp. on the Theory of Switching, Part II, p. 285.
27. Murchland J. D. (1965), A new method for finding all elementary paths in a complete directed graph, London School of Economics, Report LSE-TNT-22.
28. Murchland J. D. (1967), The once-through method of finding all shortest distances in a graph from a single origin, London Graduate School of Business Studies, Report LBS-TNT-56.
29. Nicholson T. A. J. (1966), Finding the shortest route between two points in a network, *The Computer J.*, **9**, p. 275.
30. Raimond J. F. (1969), Minimaximal paths in disjunctive graphs by direct search, *IBM J. of Res. and Dev.*, **13**, p. 391.

31. Reiter R. (1968), Scheduling parallel computation, *J. of ACM*. 15, p. 590.
32. Sakarovitch M. (1966), The *k*-shortest routes and *k*-sbortest chains in a graph, Operations Research Centre, University of California, Berkeley, Report ORC 66-32.
33. Shapiro J. F. (1968), Shortest route method for finite state space dynamic programming problems, *J. of SIAM* (*Appl. Maths.*), 16, p. 1232.
34. Taha H. A. (1971), Operations Research, Macmillan, New York.
35. Yen J. Y. (1971), Finding the *k*-shortest, loopless paths in a network, *Man. Sci.*, 17, p. 712.
36. Yen J. Y. (1971), On the efficiencies of algorithms for detecting negative loops in networks, Santa Clara Business Review, p. 52.
37. Zaloom V. (1971), On the resource-constrained project sheduling problem, *AIIE Trans.*, 3, p. 302.

Глава 9

ЦИКЛЫ, РАЗРЕЗЫ И ЗАДАЧА ЭЙЛЕРА

1. Введение

Целью настоящей главы является изучение циклов в графах и исследование некоторых из их свойств и связей с другими понятиями (такими, как понятие дерева), введенными ранее. Два типа циклов в графах эйлеровы и гамильтоновы циклы, часто встречаются в практических задачах и поэтому представляют особый интерес. Первые из них рассмотрены в настоящей главе, а вторые — в следующей.

В настоящей главе мы будем заниматься циклами как в ориентированных, так и в неориентированных графах, а также в мультиграфах, являющихся слабым обобщением понятия графа. Последние являются графами, в которых может существовать несколько дуг (x_i, x_j) между двумя данными вершинами x_i и x_j. Если наибольшее число «запараллеленных» дуг равно s, то граф называется s-графом. Так, на рисунке 9.1 представлен 3-граф. Очевидно, что обычный граф можно рассматривать как 1-граф.

s-Графы очень часто возникают на практике, когда граф используется для представления физической системы в естественных науках, вопросах управления, инженерном деле и т. д. Так, например, 3-граф на рис. 9.1 изображает молекулярную структуру органического химического соединения акрилонитрила. В электротехнике графы, изображающие электрические цепи, почти всегда являются s-графами, так как параллельно может быть включено несколько электрических компонентов. В проблемах надежности оборудования или сетей связи наиболее важные устройства или линии связи часто дублируются, утраиваются и т. д., а возникающая избыточность увеличивает надежность системы. Графы таких систем также являются s-графами.

2. Цикломатическое число и фундаментальные циклы

Пусть G — неориентированный s-граф с n вершинами, m ребрами и p связными компонентами.

Определим число $\rho(G)$ как

$$\rho(G) = n - p, \tag{9.1}$$

$\rho(G)$ дает тогда полное число ребер в остовах каждой из p связных компонент графа G.

Число $\nu(G)$ определяется как

$$\nu(G) = m - \rho(G) = m - n + p \tag{9.2}$$

и называется *цикломатическим числом* (иногда его называют также дефектом или *первым числом Бетти*), а число $\rho(G)$ называется *коцикломатическим числом*.

В теории электрических цепей (и на самом деле при представлении любой системы с сосредоточенными параметрами) числа

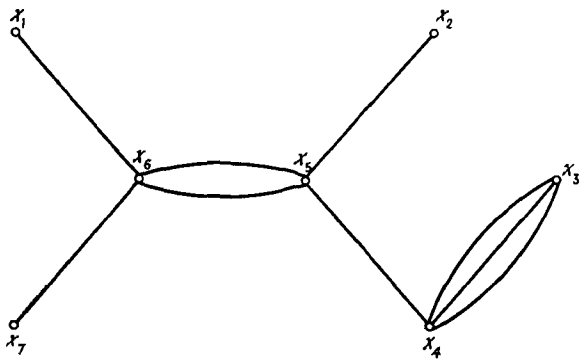


Рис. 9.1.

$\rho(G)$ и $\nu(G)$ имеют прямой физический смысл. Так, цикломатическое число равно наибольшему числу независимых контуров в графе электрической цепи, т. е. наибольшему числу независимых круговых токов, которые могут протекать в цепи. Коцикломатическое число равно наибольшему числу независимых разностей потенциалов между узлами электрической цепи.

Рассмотрим, например, электрическую цепь, показанную на рис. 9.2а. Неориентированный граф G этой цепи состоит из единственной связной компоненты ($p = 1$) и изображен на рис. 9.2б, где остов T показан жирной линией (см. гл. 6).

Добавление любого ребра (x_i, x_j) из G, не принадлежащего T, к ребрам дерева T приводит к образованию точно одного (простого) цикла, состоящего из ребер остова T, лежащих на (единственной) цепи из x_j в x_i, и только что добавленного ребра. Например, добавляя ребро $a_3 = (x_1, x_2)$, получаем цикл $(x_1, x_7, x_6, x_3, x_2, x_1)$. Так как в графе G имеется m ребер, $n - 1$ из которых лежат в T, то число всех циклов, построенных таким способом, равно $m - n + 1$, что совпадает с цикломатическим числом графа G. В примере на рис. 9.2 число таких циклов равно $15 - 7 + 1 = 9$, они изображены на рисунке пунктирными линиями и обозначены через

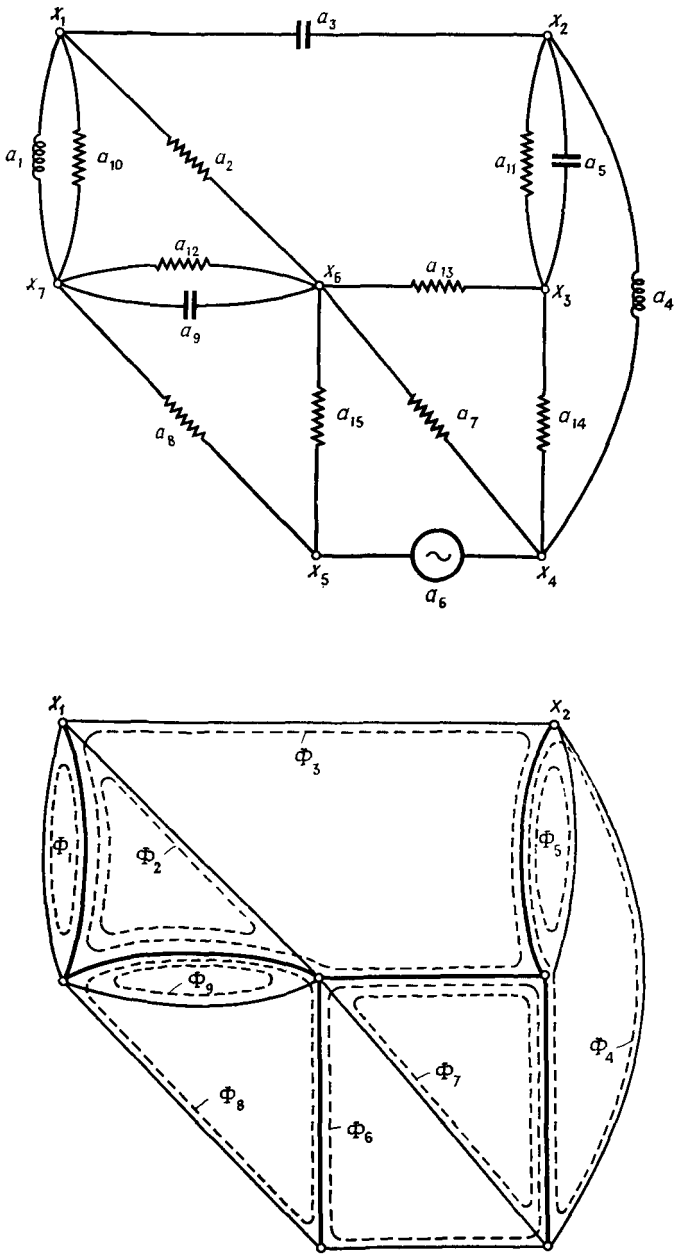


Рис. 9.2. (*а*) Электрическая цепь. (*б*) Фундаментальные циклы.

Φ_1, Φ_2, . . ., Φ_9. Все эти циклы, очевидно, независимы между собой, так как каждый из них имеет по крайней мере одно ребро, не принадлежащее никакому другому циклу. В общем случае $\nu(G)$ циклов, получаемых добавлением какого-либо ребра из G, не лежащего в T, к ребрам T, называются *фундаментальными циклами*. Если семейство этих циклов обозначить через Φ (в нашем примере $\Phi = \{\Phi_1, \Phi_2, \ldots, \Phi_9\}$), то любой другой цикл в графе, не принадлежащий к Φ, может быть выражен в виде линейной комбинации циклов из Φ, если принять следующее соглашение [21].

Пусть каждый фундаментальный цикл Φ_i, $i = 1, \ldots, \nu(G)$, представлен m-мерным вектором, в котором j-я компонента равна 1 или 0 в зависимости от того, принадлежит ли j-е ребро данному циклу. Тогда, используя $\oplus$ как символ сложения по модулю 2, любой цикл Φ_k можно представить как сумму по модулю 2 фундаментальных циклов. Так, цикл $\Phi_{10} = (a_3, a_{11}, a_{14}, a_6, a_8, a_1)$ на рис. 9.2(б) может быть представлен в виде

$$\begin{aligned}\Phi_{10} &= \Phi_1 \oplus \Phi_3 \oplus \Phi_6 \oplus \Phi_8 \\ &= (100000000100000) \oplus (001000000111100) \\ &\quad \oplus (000001000000111) \oplus (000000010001001) \\ &= (101001010010010)\end{aligned}$$

Отметим, что обращение вышеприведенного утверждения неверно, а именно некоторая сумма по модулю 2 фундаментальных циклов не обязательно дает единственный цикл, но может представлять два и более циклов. Например, сумма $\Phi_2 \oplus \Phi_3 \oplus \Phi_6 \oplus \Phi_7$ соответствует двум циклам $(a_2, a_3, a_{11}, a_{13})$ и (a_6, a_7, a_{15}). Таким образом, для порождения всех циклов графа G не надо брать все $2^{\nu(G)} - 1$ комбинаций фундаментальных циклов и складывать их по модулю 2: некоторые из этих сумм на самом деле не будут циклами Более того, если данная сумма, скажем $\Phi_i \oplus \Phi_j \oplus \Phi_k \oplus \ldots$, не порождает цикла, то нельзя отбросить другие суммы, ее содержащие, так как, складывая по модулю 2 сумму $\Phi_i \oplus \Phi_j \oplus \Phi_k \oplus \ldots$ с другой суммой, например $\Phi_\alpha \oplus \Phi_\beta \oplus \Phi_\gamma \oplus \ldots$ (которая сама может не порождать цикла), мы можем получить цикл. Для преодоления этой трудности Уэлч [26] предложил некоторый метод, позволяющий отбрасывать «некорректные» комбинации фундаментальных циклов, как только они появляются. Этот метод основан на упорядочении фундаментальных циклов в соответствии со специальным множеством правил.

Следует также заметить, что, хотя число фундаментальных циклов равно $\nu(G)$, сами эти циклы определены неоднозначно и зависят от первоначально выбранного остова T. Действительно,

в графе G можно найти множество из $\nu(G)$ независимых циклов, которые нельзя получить добавлением ребер к дереву, как это делалось выше. О таком множестве не следует говорить, что оно

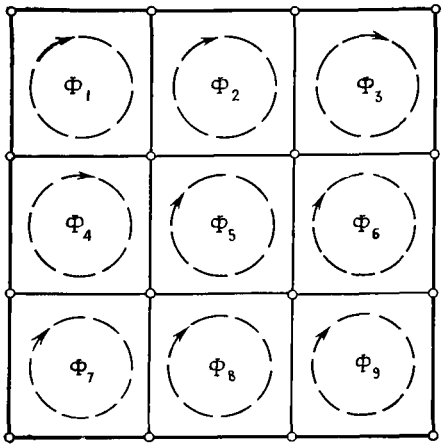


Рис. 9.3. Независимые, но нефундаментальные циклы.

«фундаментальное». На рис. 9.3 показано множество из $\nu(G) = 9$ независимых циклов графа G, которое не может быть получено добавлением ребер ни к какому остову графа G и которое поэтому не будет фундаментальным множеством.

3. Разрезы

Понятие разреза очень тесно связано с понятием цикла и определяется так.

Определение. Если вершины неориентированного графа $G = (X, A)$ разбиты на два множества X_0 и $\bar{X}_0$ (где $X_0 \subset X$ и $\bar{X}_0$ является дополнением X_0 относительно X), то множество ребер графа G, одни концевые вершины которых лежат в X_0, а другие — в $\bar{X}_0$, называется *разрезом* графа G.

Множество ребер разреза может быть представлено множеством вершин пары $(X_0, \bar{X}_0)$. Таким образом, остовный подграф $G_\mu = (X, A - (X_0, \bar{X}_0))$, полученный из G удалением ребер, принадлежащих разрезу, является несвязным графом, состоящим по крайней мере из двух компонент. В графе на рис. 9.4 множе-

ство ребер, показанных пунктиром, образует разрез, определяемый множествами $X_o = \{x_i \mid i = 1, \ldots, 4\}$ и $\tilde{X}_o = \{x_i \mid i = 5, \ldots, 11\}$. Получающийся остовный подграф состоит из четырех связных компонент:

$$\langle x_1, x_2, x_3, x_4\rangle,\ \langle x_5, x_6, x_7\rangle,\ \langle x_8\rangle \text{ и } \langle x_9, x_{10}, x_{11}\rangle.$$

Множество ребер S, удаление которых из графа G дает несвязный остовный подграф $G_p = (X, A - S)$ и при этом не существует

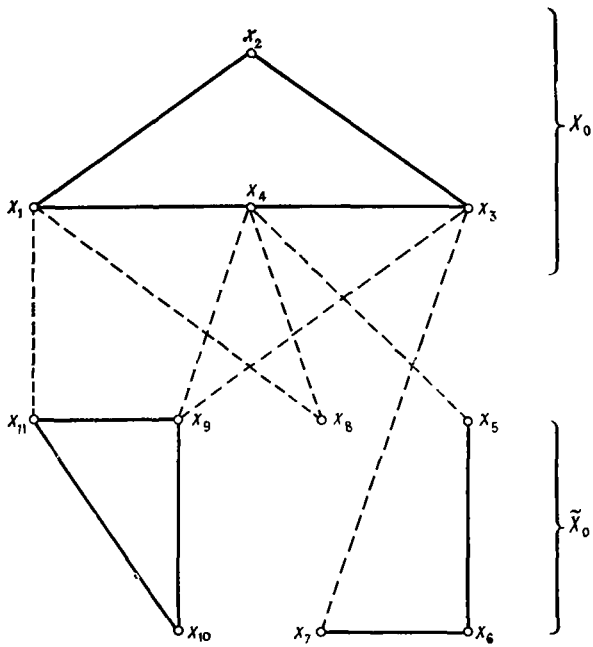


Рис. 9.4. — — — разрез $(X_o, \tilde{X}_o)$.

никакого собственного подмножества $S' \subset S$, такого, что граф $G'_p \equiv (X, A - S')$ также несвязен, называется *правильным разрезом*. Это множество можно также представить парой $(X_o, \tilde{X}_o)$, хотя теперь остовный подграф, полученный удалением правильного разреза из G, разбивает G *точно* на две связные компоненты, одна из которых имеет своими вершинами множество X_o, а другая — множество $\tilde{X}_o$. В общем случае каждый разрез является объединением некоторого числа правильных разрезов. В графе на рис. 9.4, например, разрез, показанный пунктиром, является объединением трех правильных резрезов: $\{(x_1, x_{11}), (x_4, x_9), (x_3, x_9)\}$,

$\{(x_1, x_8), (x_4, x_8)\}$ и $\{(x_3, x_7), (x_4, x_5)\}$. В остальной части этой главы, а также в следующей главе мы будем использовать слово разрез для обозначения правильного разреза, если только не оговорено противное.

Двойственность понятий остова и разреза графа G становится очевидной, если вспомнить, что остов определяется как минимальное множество ребер, связывающее все вершины графа G, в то время как разрез является минимальным множеством ребер, отделяющим одни вершины от других. Из этого замечания совершенно очевидно, что любой остов графа G должен иметь по крайней мере одно общее ребро с каждым правильным разрезом.

Разрез был определен выше для неориентированного графа. Если граф $G = (X, A)$ — ориентированный, то разрез графа G определяется как множество дуг, соответствующих ребрам неориентированного дубликата $\overline{G}$ графа G. Некоторые из дуг разреза графа G будут ориентированы из X_0 в $\tilde{X}_0$, и множество этих дуг будет записываться как $(X_0 \rightarrow \tilde{X}_0)$, в то время как множество дуг, ориентированных из вершин множества $\tilde{X}_0$ в вершины множества X_0, будет записываться как $(\tilde{X}_0 \rightarrow X_0)$. Поэтому разрез $(X_0, \tilde{X}_0)$, состоящий из дуг ориентированного графа, определяется как объединение $(X_0 \rightarrow \tilde{X}_0) \cup (\tilde{X}_0 \rightarrow X_0)$.

Например для графа на рис. 9.5 множество дуг $\{(x_2, x_5), (x_2, x_8), (x_8, x_3), (x_4, x_8), (x_7, x_4)\}$ является разрезом, разделяющим множества вершин $X_0 = \{x_1, x_2, x_3, x_4\}$ и $\tilde{X}_0 = \{x_5, x_6, x_7, x_8\}$.

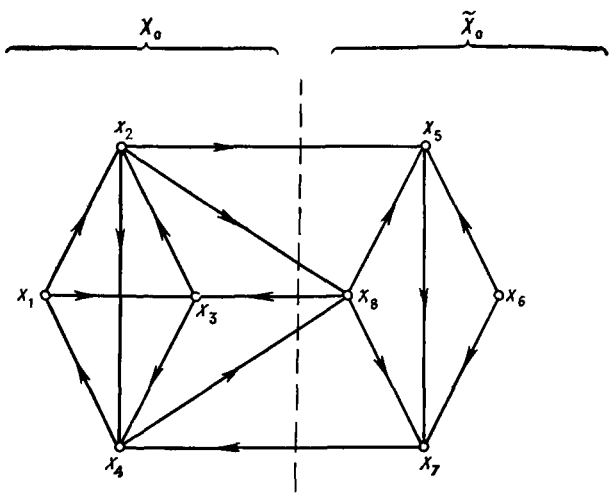


Рис. 9.5. Ориентированные разрезы.

Этот разрез образован дугами подмножеств $(X_0 \to \bar{X}_0) = \{(x_3, x_5), (x_2, x_8), (x_4, x_8)\}$ и $(\bar{X}_0 \to X_0) = \{(x_8, x_3), (x_7, x_4)\}$.

Множество фундаментальных циклов неориентированного графа G было определено в предыдущем разделе как множество $\nu(G)$ циклов, каждый из которых получается путем добавления какого-нибудь ребра, не принадлежащего остову T, к ребрам этого остова. Аналогичным образом, имея в виду отмеченную выше двойственность между разрезами и остовами, *фундаментальные разрезы* относительно остова T определяются как $n - 1$ разрезов, каждый из которых содержит одно, и только одно ребро, принадлежащее дереву T.

Следующая теорема устанавливает связь между фундаментальными разрезами и фундаментальными циклами и дает способ построения фундаментальных разрезов.

Теорема 1. *Если T — остов неориентированного графа G, то фундаментальный разрез, определяемый ребром a_i из T, образован a_i и теми ребрами из G, не принадлежащими T, которые после добавления к T дают фундаментальные циклы, содержащие a_i.*

Доказательство. Если из остова T удалить ребро a_i, то T распадается на два поддерева T_1 и T_2 (см. рис. 9.6). Любое ребро, одна концевая вершина которого лежит в T_1, а другая в T_2, должно принадлежать фундаментальному разрезу, так как добавление любого такого ребра к ребрам из T_1 и T_2 приводит к образованию другого остова графа G и, следовательно, любое множество, не содержащее таких ребер, не будет разрезом. Множество таких ребер вместе с ребром a_i является разрезом, так как их

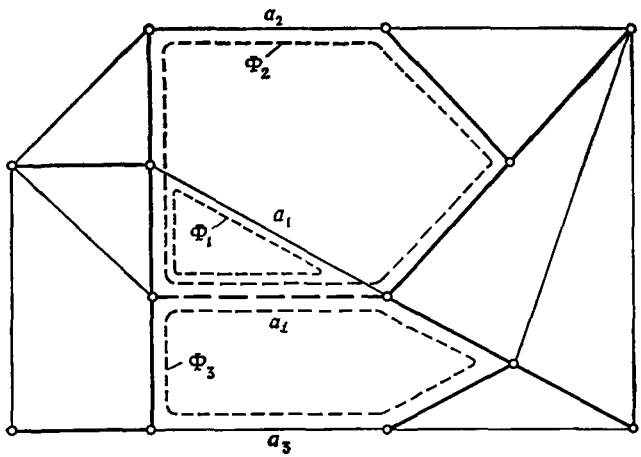


Рис. 9.6. Фундаментальные циклы, содержащие ребро a_i.

удаление разбивает граф на два подграфа, один из которых имеет множеством своих вершин T_1, а другой — T_2. Значит, этот разрез является фундаментальным разрезом. Более того, так как ребро a_i является единственным ребром, через которое проходят цепи остова T, начинающиеся в вершинах из T_1 и оканчивающиеся в вершинах из T_2, то единственными ребрами, замыкающими фундаментальные циклы, содержащие ребро a_i, будут те, одна концевая вершина которых лежит в T_1, а другая в T_2 (таковы, например, ребра a_1, a_2 и a_3 в графе на рис. 9.6). Это доказывает теорему.

4. Матрицы циклов и разрезов

Пусть T — остов неориентированного графа. *Матрицей фундаментальных циклов* графа G называется матрица $\mathbf{\Phi} = [\varphi_{ij}]$, состоящая из $\nu(G)$ строк и m столбцов, в которой φ_{ij} равно 1, если ребро a_j принадлежит циклу Φ_i, и равно 0 в противном случае. Если ребра, не принадлежащие дереву T, пронумеровать последовательно от 1 до $\nu(G)$, а ребра дерева T пронумеровать от $\nu(G) + 1$ до m, то матрица циклов $\mathbf{\Phi}$ будет иметь вид

$$\mathbf{\Phi} = (\mathbf{I} \mid \mathbf{\Phi}_{12}),$$

где $\mathbf{I}$ — единичная матрица. Это объясняется тем, что каждый цикл Φ_i содержит одно, и только одно ребро, не принадлежащее остову T, и циклы всегда можно пронумеровать по числу принадлежащих им внеостовных ребер, вследствие чего все единицы в первой $(\nu(G) \times \nu(G))$-подматрице матрицы $\mathbf{\Phi}$ лежат на диагонали.

Матрица фундаментальных разрезов $\mathbf{K} = [k_{ij}]$ определяется аналогично — как матрица с $n - 1$ строками и m столбцами, где k_{ij} есть 1, если ребро a_j принадлежит разрезу K_i, и 0 в противном случае. При той же самой нумерации ребер, что и выше, матрица $\mathbf{K}$ имеет вид

$$\mathbf{K} = (\mathbf{K}_{11} \mid \mathbf{I}),$$

так как теперь каждый фундаментальный разрез содержит одно, и только одно ребро из ребер дерева T.

В графе на рис. 9.7 возьмем остов, изображенный жирными линиями. Тогда матрица фундаментальных циклов равна

$\mathbf{\Phi} =$

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}
Φ_1	1	0	0	0	0	0	0	1	1	0	0	0	0
Φ_2	0	1	0	0	0	0	0	0	1	1	1	0	0
Φ_3	0	0	1	0	0	0	0	0	0	0	0	1	1
Φ_4	0	0	0	1	0	1	1	1	0	1	0	0	1
Φ_5	0	0	0	0	1	1	1	1	0	0	0	0	0

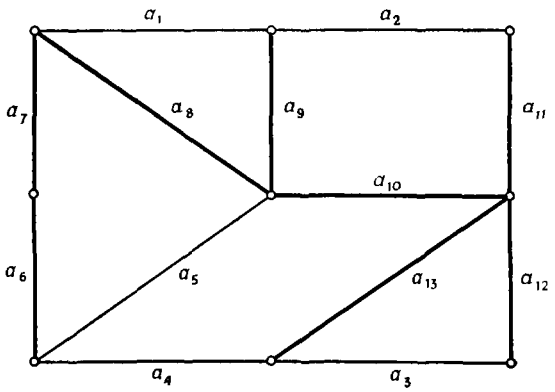


Рис. 9.7.

матрица фундаментальных разрезов равна

$$\mathbf{K} = \begin{array}{c|cccccccccccc} & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 & a_9 & a_{10} & a_{11} & a_{12} & a_{13} \\ \hline K_1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ K_2 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ K_3 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ K_4 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ K_5 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ K_6 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ K_7 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ K_8 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}$$

где разрезы K_i, соответствующие ребрам остова, имеют номера $\nu(G) + i = 5 + i$, $i = 1, 2, \ldots, 8$.

Существует несколько интересных соотношений между матрицами циклов, разрезов и инциденций. Ниже все арифметические операции являются операциями по модулю 2. Здесь мы будем рассматривать неориентированные графы без петель.

Теорема 2. *Матрица инциденций* $\mathbf{B}$ *и транспонированная матрица фундаментальных циклов* $\mathbf{\Phi}^t$ *ортогональны, т. е.* $\mathbf{B} \cdot \mathbf{\Phi}^t = 0$.

Теорема 3. *Матрица фундаментальных циклов* $\mathbf{\Phi}$ *и транспонированная матрица фундаментальных разрезов* $\mathbf{K}^t$ *ортогональны, т. е.* $\mathbf{\Phi} \cdot \mathbf{K}^t = 0$.

Эти две теоремы являются немедленным следствием двух очевидных фактов:

(1) Каждая вершина в цикле инцидентна четному числу ребер (а в случае простого цикла — двум ребрам) этого цикла.

(2) Каждый разрез цикла, индуцированный некоторым разрезом, имеет четное число ребер, общих с этим разрезом.

Теорема 2 следует из (1), а теорема 3 — из (2), если вспомнить, что все операции рассматриваются по модулю 2, так что, например, $2 \equiv 0 \pmod 2$. Кроме того, существуют доказательства этих теорем [21, 17, 6], не использующие предположения о фундаментальности циклов и разрезов. Следовательно, сформулированные теоремы справедливы для любых матриц циклов и разрезов. Последние определяются аналогичным образом.

По теореме 3 можно написать

$$\mathbf{\Phi K}^t = (\mathbf{I} \mid \mathbf{\Phi}_{12}) \cdot \begin{pmatrix} \mathbf{K}_{11}^t \\ \hline \mathbf{I} \end{pmatrix} = \mathbf{K}_{11}^t + \mathbf{\Phi}_{12} = 0.$$

Поэтому

$$\mathbf{K}_{11}^t = -\mathbf{\Phi}_{12} = \mathbf{\Phi}_{12}, \tag{9.3}$$

так как $-1 = 1 \pmod 2$. Вследствие этого матрица фундаментальных разрезов может быть немедленно получена, как только известна матрица фундаментальных циклов, и наоборот. Справедливость соотношения (9.3) можно проверить на приведенных выше матрицах $\mathbf{\Phi}$ и $\mathbf{K}$ для графа, изображенного на рис. 9.7.

5. Эйлеровы циклы и задача китайского почтальона

Определение. Если дан неориентированный s-граф G, то *эйлеров цикл* (эйлерова цепь) — это такой цикл (цепь), который проходит ровно один раз по каждому ребру.

Очевидно, не все графы имеют эйлеровы циклы (см., например, граф на рис. 9.8), но если эйлеров цикл существует, то это озна-

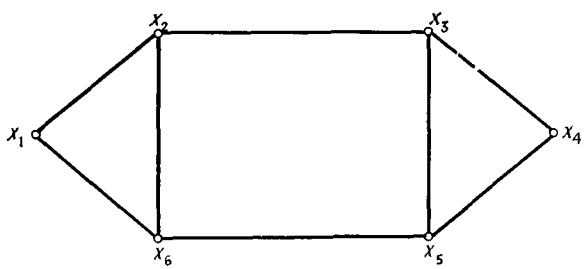


Рис. 9.8. Граф без эйлерова цикла.

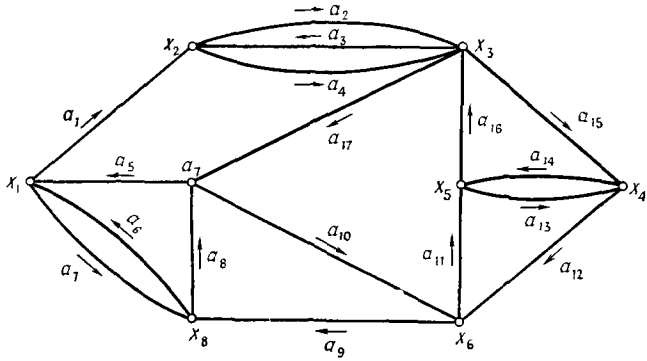


Рис. 9.9.

чает, что, следуя вдоль этого цикла, можно нарисовать граф на бумаге, не отрывая от нее карандаша. 3-граф на рис. 9.9 имеет следующий эйлеров цикл (начиная с вершины x_1):

$$a_1a_2a_3a_4a_{15}a_{14}a_{13}a_{12}a_{11}a_{16}a_{17}a_{10}a_9a_8a_5a_7a_6.$$

Направление движения по каждому ребру показано на рисунке стрелками.

Эйлер первым в своей знаменитой *задаче о Кёнигсбергских мостах* рассмотрел вопрос о существовании таких циклов в графах.

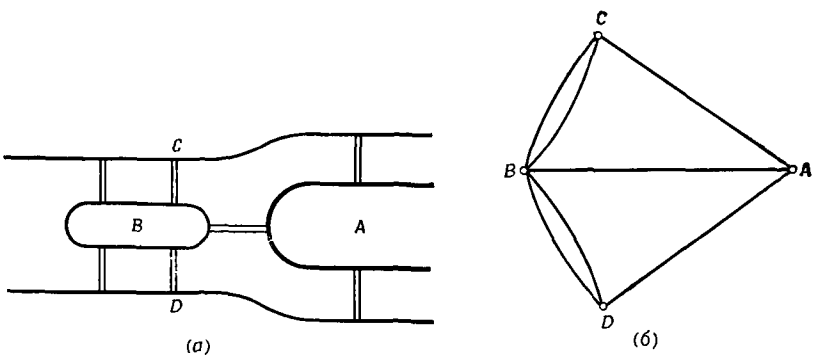


Рис. 9.10. (*а*) Карта Кёнигсберга. (*б*) Эквивалентный граф.

Кёнигсберг (теперь Калининград) расположен на обоих берегах реки Преголя и на двух островах этой реки. Берега реки и два острова соединены семью мостами как показано на карте на рис. 9.10 (а).

Вопрос — поставленный в 1736 году — состоит в том, можно ли, начав с некоторой точки, совершить прогулку и вернуться

в исходную точку, пройдя по каждому мосту ровно один раз. Если каждый берег реки и острова считать вершиной графа, а каждый мост — ребром, то карту рис. 9.10 (а) можно представить в виде 2-графа, изображенного на рис. 9.10 (б), и ответ на поставленный вопрос зависит теперь от существования эйлерова цикла в этом графе. Эйлер установил, что указанный граф не содержит эйлерова цикла, и этот результат ознаменовал рождение теории графов.

Основная теорема о существовании эйлеровых циклов формулируется так.

Теорема 4 (а). *Связный неориентированный s-граф G содержит эйлеров цикл (эйлерову цепь) тогда и только тогда, когда число вершин нечетной степени равно* 0 (0 *или* 2).

Доказательство. Мы докажем эту теорему для случая цикла; случай цепи рассматривается аналогично.

Необходимость. Любой эйлеров цикл должен прийти в вершину по одному ребру и покинуть ее по другому, так как любое ребро должно использоваться ровно один раз. Поэтому, если G содержит эйлеров цикл, то степени всех вершин должны быть четными.

Достаточность. Пусть G — связный неориентированный s-граф, все вершины которого имеют четную степень. Начнем путь из некоторой произвольной вершины x_0 и пойдем по некоторому ранее не использованному ребру к следующей вершине, и так до тех пор, пока не вернемся в вершину x_0 и не замкнем цикл. Если все ребра окажутся использованными, то нужный эйлеров цикл построен. Если же некоторые ребра не использованы, то пусть Ф — только что построенный цикл. Так как граф G связен, то цикл Ф должен проходить через некоторую вершину, скажем x_i, являющуюся конечной вершиной какого-либо до сих пор неиспользованного ребра. Если удалить все ребра, принадлежащие Ф, то в оставшемся графе все вершины по-прежнему будут иметь четную степень, так как в цикле Ф должно быть четное число ребер (0 является четным числом), инцидентных каждой вершине.

Начиная теперь с x_i, получаем цикл Ф′, начинающийся и оканчивающийся в x_i. Если все оставшиеся ранее ребра использованы для цикла Ф′, то процесс окончен. Нужный эйлеров цикл будет образован частью цикла Ф от вершины x_0 до x_i, затем циклом Ф′ и, наконец, частью цикла Ф от вершины x_i до x_0. Если же все еще остались неиспользованные ребра, то объединение построенных выше циклов Ф и Ф′ дает новый цикл Ф. Мы снова можем найти вершину x_j, принадлежащую циклу и являющуюся концевой вершиной некоторого неиспользованного ребра. Затем мы можем приступить к построению нового цикла Ф′, начинающегося в x_j, и так продолжать до тех пор, пока не будут использованы все ребра и не будет получен таким образом эйлеров цикл Ф. Это доказывает теорему.

Очевидно, что если граф G несвязен (за исключением изолированных вершин), то эйлеров цикл не существует, так как нет никакой цепи, ведущей из одной его компоненты в другую. Очевидно также, что в случае эйлеровой цепи две ее конечные вершины p и q являются вершинами нечетной степени.

Если G — ориентированный s-граф, то справедлива теорема, полностью аналогичная предыдущей, а именно

Теорема 4 (б). *Связный ориентированный s-граф G содержит эйлеров цикл (эйлерову цепь) тогда и только тогда* [1]*, когда все полустепени захода $d_t(x_i)$ и полустепени исхода $d_o(x_i)$ вершин удовлетворяют условиям:*

$$\text{для случая цикла} \quad d_t(x_i) = d_0(x_i,)\, \forall x_i \in X,$$

$$\text{для случая цепи} \quad d_t(x_i) = d_o(x_i)\, \forall x_i \neq p \quad \text{или} \quad q,$$

$$d_t(q) = d_o(q) + 1$$

$$\text{и } d_t(p) = d_o(p) - 1,$$

где p — начальная, а q — конечная вершины эйлеровой цепи.

Флёри [16] дал очень простой алгоритм построения эйлерова цикла в неориентированном графе (если такой цикл существует). Этот алгоритм легко может быть распространен на ориентированные графы (см. также задачи 7 и 8) и состоит в следующем. *Алгоритм нахождения эйлерова цикла.* Начинать с некоторой вершины p и каждый раз вычеркивать пройденное ребро. Не проходить по ребру, если удаление этого ребра приводит к разбиению графа на две связные компоненты (не считая изолированных вершин).

5.1. Некоторые родственные задачи

Сразу же вспоминается ряд вопросов, связанных с тем, имеется ли в неориентированном графе эйлеров цикл. Например:

(i) Каково наименьшее число цепей или циклов необходимо для того, чтобы каждое ребро графа G содержалось точно в одной цепи или в одном цикле? Очевидно, что если G имеет эйлерову цепь или эйлеров цикл, то ответом будет число один [2].

(ii) Ребрам графа G приписаны положительные веса. Требуется найти цикл, проходящий через каждое ребро графа G *по крайней мере один раз* и такой, что для него общий вес (а именно сумма величин $n_j c(a_j)$, где число n_j показывает, сколько раз проходи-

[1]) В этой теореме, естественно, циклы и цепи — ориентированные.— *Прим. ред.*

[2]) Этот вопрос относится к нахождению наименьшего числа реберно непересекающихся подграфов (в случае цепей или циклов), необходимых для «покрытия» ребер графа G. Задачи этого типа обсуждались также в гл. 3 как варианты *задачи о покрытии множества.*

лось ребро a_j, а $c(a_j)$ — вес ребра) минимален. Очевидно, что если G содержит эйлеров цикл, то любой такой цикл будет оптимальным, так как каждое ребро проходится только один раз и вес этого цикла равен тогда $\sum_{j=1}^{m} c(a_j)$.

Сформулированная выше задача (ii) называется задачей *китайского почтальона* [18], и ее решение имеет много потенциальных приложений, как например:

(А) **Сбор мусора.** Рассмотрим проблему сбора домашнего мусора. Допустим, что определенный район города обслуживается единственной машиной. Ребра графа G представляют дороги, а вершины — пересечения дорог. Величина $c(a_j)$ — вес ребра — будет соответствовать длине дороги. Тогда проблема сбора мусора в данном районе сводится к нахождению цикла в графе G, проходящего по каждому ребру G по крайней мере один раз. Требуется найти цикл с наименьшим километражем. В действительности емкость машины и продолжительность рабочего дня накладывают ограничения на число улиц, которые может обслужить одна машина за день. То, что действительно может требоваться,— это не нахождение отдельного цикла, а некоторого числа Q циклов, обслуживаемых по одному в день, скажем, за период в Q дней; при этом циклы могут быть выбраны так, что не нарушаются вышеупомянутые ограничения [1, 2, 4, 11, 22].

(Б) **Доставка молока или почты.** Еще две задачи, когда требуется определить маршрут, проходящий хотя бы один раз по каждой из улиц, возни[illegible] при доставке молока или почты. Здесь задача состоит в нахождении маршрута, минимизирующего общий километраж (или время, стоимость и т. д.).

(В) **Проверка электрических, телефонных или железнодорожных линий.** Проблема инспектирования распределенных систем (лишь некоторые из которых названы выше) связана с непременным требованием проверки всех «компонент». Поэтому она также является проблемой типа (ii) или близка к ней.

Другие приложения могут быть связаны с разбрасыванием смеси песка и соли на главных дорогах зимой для предотвращения обледенения, с наилучшим методом работы автоматических вентиляционных устройств, с уборкой помещений и коридоров в больших учреждениях и даже с наилучшим маршрутом осмотра музея!

5.2. Алгоритм для задачи китайского почтальона

В случае когда веса всех ребер равны единице, задачу китайского почтальона рассмотрели Беллман и Кук [3] с использованием динамического программирования. Более общую задачу,

когда веса ребер произвольны, сформулировали и решили как задачу о паросочетании (с частной задачей о кратчайшей цепи) Эдмондс [9], Эдмондс и Джонсон [10], Басакер и Саати [7] и Кристофидес [8].

Рассмотрим неориентированный граф $G = (X, A)$. Среди вершин из X некоторые вершины (скажем из множества X^+) будут иметь четные степени, а остальные (из множества $X^- \equiv X - X^+$) — нечетные степени. Далее сумма степеней d_i всех вершин $x_i \in X$ равна удвоенному числу ребер в A (так как каждое ребро добавляет по единице к степеням двух его концевых вершин) и поэтому равна четному числу $2m$. Следовательно,

$$\sum_{x_i \in X} d_i = \sum_{x_i \in X^+} d_i + \sum_{x_i \in X^-} d_i = 2m,$$

и так как сумма $\sum_{x_i \in X^+} d_i$ четна, то сумма $\sum_{x_i \in X^-} d_i$ также четна. Но все d_i в последней сумме нечетны, значит число $|X^-|$ вершин нечетной степени четно.

Пусть M — множество таких цепей (скажем μ_{ij}) в G между концевыми вершинами x_i и x_j ($x_i, x_j \in X^-$), что никакие две цепи не имеют одинаковых конечных вершин, т. е. цепи соединяют различные пары вершин из X^- и покрывают все вершины [1]) множества X^-. Число цепей μ_{ij} в M равно $\frac{1}{2}|X^-|$, а как было показано выше, $|X^-|$ всегда четно, следовательно, это число всегда целое, если, конечно, оно определено. Предположим теперь, что все ребра, образующие цепь μ_{ij}, добавлены к G в качестве *искусственной* параллельной цепи, причем эти ребра остаются в графе G. Это означает, прежде всего, что все ребра из G, образующие цепь μ_{ij}, теперь удвоены. Так поступаем с каждой цепью $\mu_{ij} \in M$, и полученный s-граф обозначим через $G^-(M)$. Так как некоторые ребра из G могут входить более чем в одну цепь μ_{ij}, то некоторые ребра из $G^-(M)$ могут быть (после того как добавлены все «новые» цепи μ_{ij}) утроены, учетверены и т. д.

Теперь мы можем доказать следующую теорему.

Теорема 5. *Для любого цикла, проходящего по G, можно выбрать множество M, для которого граф $G^-(M)$ имеет эйлеров цикл, соответствующий первоначально взятому циклу в графе G. Это соответствие таково, что если цикл проходит по ребру (x_i, x_j) из G l раз, то в $G^-(M)$ существует l ребер (одно реальное и $l-1$ искусственных) между x_i и x_j, каждое из которых проходится ровно один раз эйлеровым циклом из $G^-(M)$. Справедливо и обратное утверждение.*

[1]) В дальнейшем такое множество цепей будем называть *цепным паросочетанием* (на множестве X^- или для множества X^-).— *Прим. ред.*

Доказательство. Если цикл проходит по графу G, то по крайней мере одно ребро, инцидентное каждой вершине x_i нечетной степени, должно проходиться дважды. (Ребро, проходимое дважды, можно рассматривать как два параллельных ребра — одно реальное и одно искусственное — и каждое из них проходится один раз.) Пусть это ребро (x_i, x_k). В случае нечетности степени d_k вершины x_k графа G добавление искусственного ребра прежде всего сделает d_k четным, и значит только ребро (x_i, x_k) нужно будет проходить дважды, если ограничиться рассмотрением лишь вершин x_i и x_k. В случае когда d_k четно, добавление искусственного ребра сделает d_k нечетным, а второе ребро выходящее из x_k, должно быть пройдено дважды (т. е. добавляется еще одно искусственное ребро). Такая ситуация сохраняется до тех пор, пока не встретится вершина нечетной степени, о чем говорилось выше. Следовательно, чтобы удовлетворить условию возвращения в вершину x_i, нужно дважды пройти всю цепь из x_i в некоторую другую вершину нечетной степени x_r, принадлежащую множеству X^-. Это автоматически приводит к выполнению условия прохождения вершины x_r. Аналогично обстоит дело для всех других вершин x_i из X^-. Это значит, что все множество M цепей из G, определенное выше, должно проходиться дважды, и так как отсюда вытекает, что каждое ребро из $G^-(M)$ должно проходиться один раз, то теорема доказана.

Алгоритм решения задачи китайского почтальона немедленно следует из доказанной теоремы, так как все, что для этого необходимо, состоит в нахождении множества цепей M^* (цепного паросочетания для множества вершин нечетной степени), дающего наименьший дополнительный вес. Цикл наименьшего веса, проходящий по G, будет иметь вес, равный сумме весов всех ребер из G плюс сумма весов ребер в цепях из M^*. Это то же самое, что и сумма весов всех ребер — реальных и искусственных — графа $G^-(M^*)$. Дадим теперь описание алгоритма.

5.2.1. Описание алгоритма. *Шаг* 1. Пусть $[c_{ij}]$ — матрица весов ребер графа G. Используя алгоритм кратчайшей цепи (см. гл. 8), образуем $|X^-| \times |X^-|$-матрицу $\mathbf{D} = [d_{ij}]$, где d_{ij} — вес цепи наименьшего веса, идущей из некоторой вершины $x_i \in X^-$ в другую вершину $x_j \in X^-$.

Шаг 2. Найдем то цепное паросочетание M^* для множества X^-, которое дает наименьший вес (в соответствии с матрицей весов $\mathbf{D}$). Это можно сделать эффективно с помощью алгоритма минимального паросочетания из гл. 12.

Шаг 3. Если вершина x_α сочетается с другой вершиной x_β, то определим цепь $\mu_{\alpha\beta}$ наименьшего веса (из x_α в x_β), соответствующую весу $d_{\alpha\beta}$, делая шаг 1. Добавим искусственные ребра в G,

соответствующие ребрам из $\mu_{\alpha\beta}$, и проделаем это для всех других цепей из множества M^*, в результате чего получится s-граф $G^-(M^*)$.

Шаг 4. Сумма весов всех ребер графа $G^-(M^*)$, найденная с использованием матрицы $[c_{ij}]$ (вес искусственного ребра берется равным весу параллельного ему реального ребра), равна минимальному весу цикла, проходящего по G. При этом число прохождений цикла по ребру (x_i, x_j) равно общему числу параллельных ребер между x_i и x_j в $G^-(M^*)$.

Следует заметить, что поскольку на шаге 2 мы используем минимальное паросочетание, никакие две кратчайшие цепи μ_{ij} и μ_{pq} при таком паросочетании (скажем, идущие из x_i в x_j и из x_p

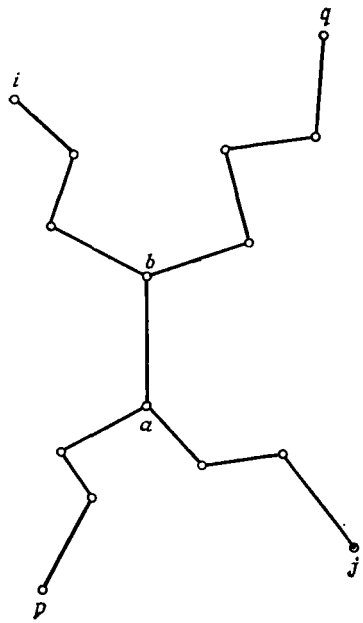


Рис. 9.11. Цепи μ_{ij} и μ_{pq}, имеющие общее ребро (a, b).

в x_q) не могут иметь общего ребра. Если бы они имели общее ребро (x_a, x_b), как показано на рис. 9.11, то сочетание вершин x_i и x_q (использующее подцепи от x_i к x_b и от x_b к x_q) и сочетание пары вершин x_p и x_j (использующее подцепи от x_p к x_a и от x_a к x_j) давало бы общее паросочетание веса $2c_{ab}$, меньшего чем вес первоначального паросочетания, что противоречит предположению о минимальности исходного паросочетания. Следовательно, граф $G^-(M^*)$ не должен содержать более двух параллельных ребер

между любыми двумя вершинами, т. е. оптимальный цикл не проходит ни по какому ребру графа G более чем два раза.

5.2.2. Пример. Рассмотрим граф G, изображенный на рис. 9.12, имеющий 12 вершин и 22 ребра, веса которых указаны у каждого ребра. Задача состоит в нахождении цикла, проходящего

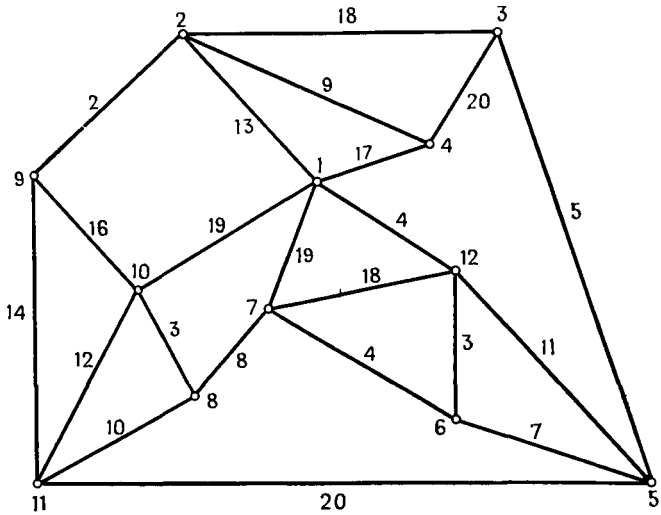


Рис. 9.12. Граф из примера 5.2.2.

по каждому ребру G по крайней мере один раз и имеющего минимальный вес. Множество вершин нечетной степени этого графа есть {1, 3, 4, 6, 8, 9}. Используя алгоритм кратчайшей цепи Дейкстры (см. гл. 8), найдем матрицу **D** из шага 1:

D =

	1	3	4	6	8	9
1	—	19	17	7	19	15
3	19	—	20	12	24	20
4	17	20	—	24	30	11
6	7	12	24	—	12	22
8	19	24	30	12	—	19
9	15	20	11	22	19	—

На шаге 2 алгоритм минимального паросочетания (см. гл. 12) связывает следующие вершины (два других паросочетания также являются минимальными):

1 с 6, цепь 1—12—6, вес 7,
3 с 8, цепь 3—5—6—7—8, вес 24,
9 с 4, цепь 9—2—4, вес 11.

Граф $G^-(M^*)$, получающийся после выполнения шага 3, изображен на рис. 9.13; пунктиром показаны искусственные ребра.

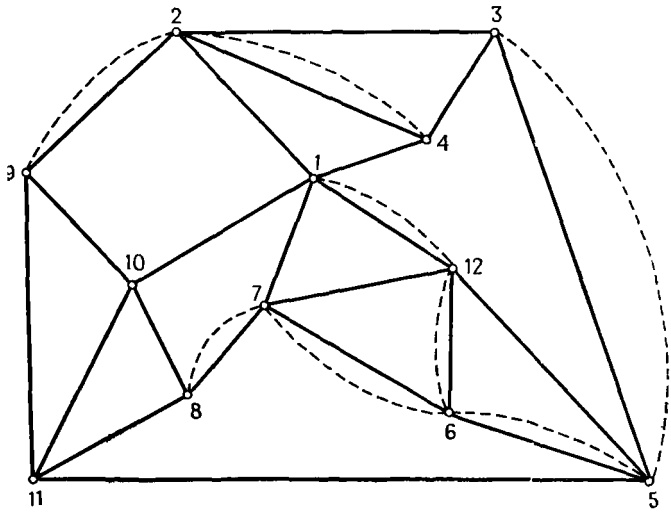


Рис. 9.13. Граф $G^-(M^*)$ из примера 5.2.2.
——— реальные ребра; — — — искусственные ребра.

Из этого рисунка видно, что оптимальный цикл в графе G проходит через ребра (9, 2), (2, 4), (3, 5), (5, 6), (1, 12), (12, 6), (7, 6) и (8, 7) дважды, а через все остальные ребра — один раз. Вес этого цикла равен 294.

Как только найден граф $G^-(M^*)$, могут быть сразу же найдены по алгоритму Флёри, упомянутому ранее, эйлеров цикл этого графа и соответствующий оптимальный цикл, проходящий по первоначальному графу G. Для графа, изображенного на рис. 9.13, например, один из возможных эйлеровых циклов, полученных по вышеприведенному правилу, дается последовательностью вершин (начиная с вершины 1):

1, 4, 3, 5, 3, 2, 4, 2, 1, 12, 5, 6, 7, 8, 10, 1, 7, 6,
12, 6, 5, 11, 9, 2, 9, 10, 11, 8, 7, 12, 1.

Этот цикл является также соответствующим оптимальным циклом, проходящим по первоначальному графу из рис. 9.12. Возможны также другие оптимальные циклы, проходящие ребра графа в ином порядке. Однако эти циклы будут проходить дважды по тому же множеству из 8 ребер. Таким образом, как только определены те ребра графа G, по которым оптимальный цикл проходит дважды, можно построить несколько таких циклов в графе G.

5.3. Связь между эйлеровыми и гамильтоновыми циклами

Гамильтонов цикл в графе G был определен в главе 1 как простой цикл, проходящий (один, и только один раз) через каждую вершину графа G. Не удивительно, что двойственность между эйлеровыми и гамильтоновыми циклами (замена вершины на ребро и наоборот) приводит к тесной связи между этими двумя понятиями в применении к неориентированному графу G и соответствующему ему *реберному графу*, определяемому ниже.

Определение. Реберный [1]) граф G_l графа G имеет столько же вершин, сколько ребер у графа G. Ребро между двумя вершинами графа G_l существует тогда и только тогда, когда ребра графа G, соответствующие этим двум вершинам, смежны (т. е. инцидентны одной и той же вершине графа G).

Например, для графа, показанного на рис. 9.14(а), граф G_l изображен на рис. 9.14(б).

Легко можно показать [15], что верны два следующие утверждения о взаимоотношении между эйлеровыми и гамильтоновыми циклами.

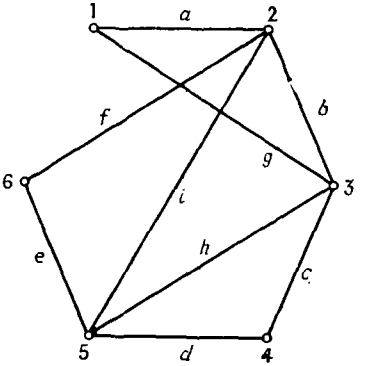
(1) Если G имеет эйлеров цикл, то G_l имеет как эйлеров, так и гамильтонов циклы.

(2) Если G имеет гамильтонов цикл, то G_l также имеет гамильтонов цикл.

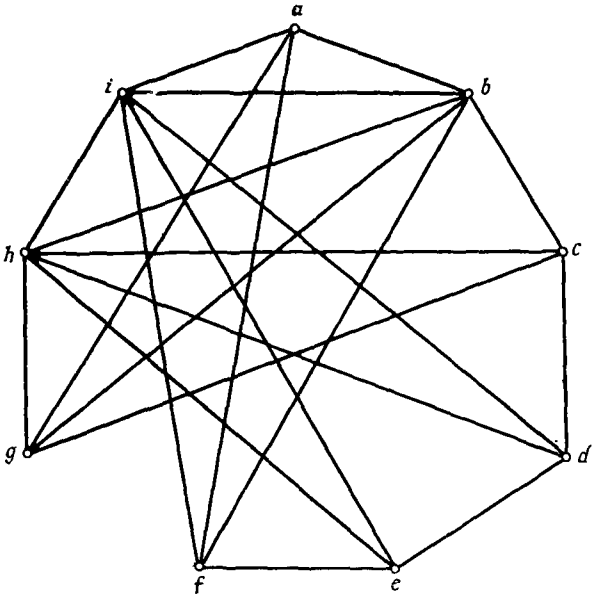
Обращение этих утверждений, как нетрудно продемонстрировать, неверно. Для графа, изображенного на рис. 9.14(а), степени всех его вершин четны и поэтому существует эйлеров цикл. Таким образом, граф G_l, изображенный на рис. 9.14(б), также имеет эйлеров цикл (поскольку степени всех его вершин опять четны) и, кроме того, имеет гамильтонов цикл, задаваемый последовательностью вершин $a, g, c, d, e, f, b, h, i, a$. Если теперь из графа G удалить ребро b, то новый граф G' не имеет эйлерова цикла, но все еще имеет гамильтонов цикл 1, 2, 6, 5, 4, 3, 1. Реберный граф G'_l графа G' является тогда графом из рис. 9.14 (б) с удаленной вершиной b (и без ребер, инцидентных вершине b). Этот граф все еще имеет гамильтонов цикл $a, f, e, d, c, g, h, i, a$.

---

[1]) Реберный граф графа G часто обозначается через $L(G)$.— *Прим. ред.*



(а)



(б)

Рис. 9.14. (*а*) — граф G. (*б*) — реберный граф G_l.

Задача выяснения существования у данного неориентированного графа G гамильтонова цикла, а также нахождения такого цикла, если он существует, является важной задачей теории графов как с практической, так и с теоретической точек зрения. Если ребрам графа G приписаны некоторые веса $[c_{ij}]$ и граф G

содержит несколько гамильтоновых циклов, то большой интерес представляет также задача нахождения такого цикла с минимальным общим весом. Ввиду важности этих вопросов гамильтоновы циклы рассматриваются отдельно в следующей главе.

6. Задачи

1. Для неориентированного графа, изображенного на рис. 9.15, найти цикломатическое и коцикломатическое числа, а также фундаментальные циклы относительно некоторого остовного дерева.

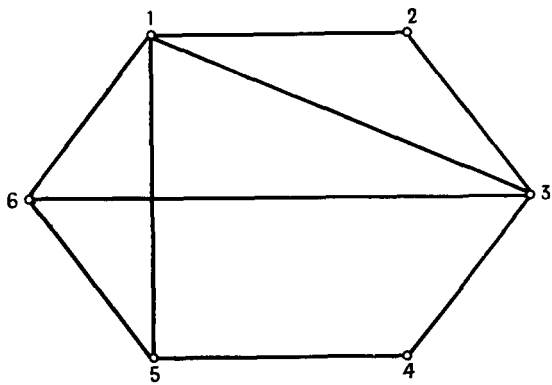


Рис. 9.15.

2. Для графа, изображенного на рис. 9.15, выписать соответственно матрицы **Ф** и **К** фундаментальных циклов и фундаментальных разрезов и на этом примере проверить справедливость теорем 2 и 3.

3. Используя теорему 2, получить выражение для матрицы фундаментальных циклов в терминах матрицы инциденций **В**.

4. Доказать, что множество K ребер неориентированного графа, такое, что K имеет четное число ребер, общих с каждым циклом, является разрезом (не необходимое свойство).

5. Доказать, что неориентированный граф G имеет эйлеров цикл тогда и только тогда, когда G является объединением реберно непересекающихся циклов.

6. Если столбцы матриц **В**, **Ф** и **К** записаны с использованием одного и того же порядка ребер (относительно некоторого остовного дерева, по которому образованы фундаментальные циклы и разрезы) и затем представлены в виде $\mathbf{B} = [\mathbf{B}_{11}, \mathbf{B}_{12}]$, $\mathbf{Ф} =$

$= [\mathbf{I}, \ \mathbf{\Phi}_{12}]$ и $\mathbf{K} = [\mathbf{K}_{11}, \ \mathbf{I}]$, то

$$\mathbf{K}_{11} = \mathbf{\Phi}_{12}^{t} = \mathbf{B}_{12}^{-1} \cdot \mathbf{B}_{11}, \qquad \mathbf{\Phi} = [\mathbf{\Phi}_{12}^{t}, \ \mathbf{I}] = \mathbf{B}_{12}^{-1} \cdot \mathbf{B}.$$

Следовательно, по любой из трех матриц **B**, **Φ** и **K** можно найти остальные.

7. Показать, что алгоритм Флёри всегда позволяет найти эйлеров цикл графа, если таковой существует.

8. Написать алгоритм, основанный на алгоритме Флёри и позволяющий найти *все* эйлеровы циклы графа.

9. Показать, что для ориентированного графа $G = (X, A)$ с полустепенями $d_0(x_i) = d_t(x_i)$, $\forall x_i \in X$, число различных эйлеровых циклов дается выражением

$$\Delta_r \cdot \prod_{x_i \in X} (d_0(x_i) - 1)!,$$

где Δ_r — число ориентированных остовных деревьев графа G с корнем x_r. Эта величина не зависит от выбора x_r (см. [24]).

10. Используя понятие эйлерова цикла, построить алгоритм обхода лабиринта от начальной точки s до конечной точки t.

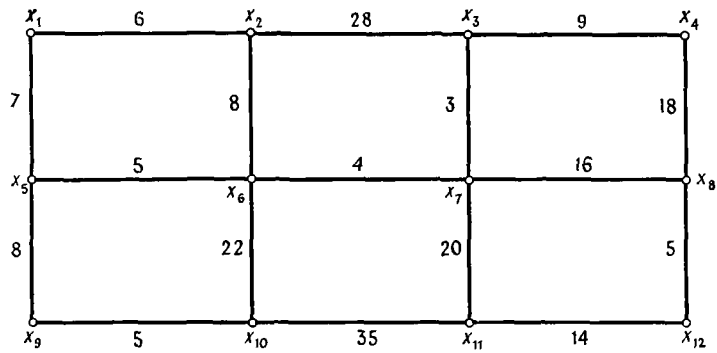


Рис. 9.16.

не проходящий ни по какому коридору дважды в одном направлении. Предполагается, что при обходе лабиринта пройденный путь не запоминается (память у путешественника отсутствует). Разрешается только использовать два типа меток (например, 0 и 1), чтобы помечать ими входы и выходы коридоров. (См. [23], [13] и [14]).

11. Решить задачу китайского почтальона для взвешенного неориентированного графа, показанного на рис. 9.16. (Воспользоваться алгоритмом гл. 8 для нахождения кратчайшей цепи и решить задачу о паросочетании с помощью полного перебора.)

7. Список литературы

1. Altman S., Beltrami E., Rappaport S., Schoepfle G. (1971), A nonlinear programming model for household refuse collection, *IEEE Trans.*, **SMC-1**, p. 289.
2. Altman S., Bhagat N., Bodin L. (1971), Extensions of the Clarke and Wright algorithm for routing garbage trucks, Presented at the 8th International TIMS Meeting, Washington D.C.
3. Bellman R., Cooke K. L. (1969), The Konigsberg bridges problem generalized, *J. of Math. Anal. and Appl.*, 25, p. 1.
4. Beltrami E. J., Bodin L. D. (1972), Networks and vehicle routing for municipal waste collection, Proc. 10th Annual Allerton Conference on Circuit and Systems Theory, University of Illinois, p. 143.
5. Berge C., Ghouila-Houri A. (1965), Programming, game and transportation networks, Methuen, London.
6. Берж К., Теория графов и ее применения, М., ИЛ., 1962.
7. Басакер Р., Саати Т., Конечные графы и сети, М., Наука, 1974.
8. Christofides N. (1973), The optimum traversal of a graph, *Omega- The Int. J. of Management Science*, 1, p. 1.
9. Edmonds J. (1965), The Chinese postman's problem, Bulletin of the Operations Research Soc. of America, 13, Supplement 1, p. B-73.
10. Edmonds J., Johnson E. (in press), Matching, Euler tours and the Chinese postman's problem.
11. Eilon S., Watson-Gandy, Christofides N. (1971), Distribution Managemenet:Mathematical modelling and practical analysis, Griffin, London.
12. Even S. (1973), Algorithmic Combinatorics, Macmillan, New York.
13. Fraenkel A. S. (1970), Economic traversal of labyrinths, *Mathematical Magazine*, 43, p. 125.
14. Fraenkel A. S. (1971), Economic traversal of labyrinths (Correction), *Mathematical Magazine*, 44, p. 1.
15. Harary F., Nash-Williams C. St. J. A. (1965), On Eulerian and Hamiltonian graphs and line graphs, *Canadian Mathematical Bulletin*, 8, p. 701.
16. Kaufmann A. (1967), Graphs, dynamic programming and finite games, Academic Press, New York.
17. Kim W. H., Chien R.T.-W. (1962), Topological analysis and synthesis of communication networks, Columbia University Press, New York.
18. Kwan M.-K. (1962), Graphic programming using odd or even points, Chinese Mathematics, 1, 273.
19. Lovasz L. (1968), On covering of graphs, Theorie des Graphs, Dunod, Paris, p. 231.
20. Marshall C. W. (1971), Applied graph theory, Wiley, New York.
21. Seshu S., Reed M. B. (1961), Linear graphs and electrical networks, Addison-Wesley, Reading, Massachusetts.
22. Stricker R. (1970), Public sector vehicle routing — the Chinese postman problem, Ph. D. Thesis, Electrical Eng. Dept., M.I.T.
23. Tarry G. (1895), Le problème des labyrinths, *Nouvelles Ann. de Math.*, 14, p. 187.
24. Van Aardenne-Ehrenfest T., de Bruijn N. G. (1951), Circuits and trees in oriented linear graphs, *Simon Stevin*, 28, p. 203.
25. Voss H. J. (1968), Some properties of graphs containing k independent circuits, Theorie des Graphes, Dunod, Paris, p. 231.
26. Welch J. T. (1966), A mechanical analysis of the cyclic structure of indirected linear graphs, *Jl. of ACM*, 13, p. 205.

Глава 10

ГАМИЛЬТОНОВЫ ЦИКЛЫ, ЦЕПИ И ЗАДАЧА КОММИВОЯЖЕРА

1. Введение

В ряде отраслей промышленности, особенно химической и фармацевтической, возникает следующая основная задача планирования. Нужно произвести ряд (скажем n) продуктов, используя единственный тип аппаратуры или реактор. Аппарат (реактор) должен или не должен быть перенастроен (очищен) после того как произведен продукт p_i (но до того, как началось производство продукта p_j), в зависимости от комбинации (p_i, p_j). Стоимость перенастройки аппаратуры постоянна и не зависит от продукта, который только что произведен, или от продукта, следующего за ним. Разумеется, не требуется никаких затрат, если перенастройка аппаратуры не нужна [1]). Предположим, что эти продукты производятся в непрерывном цикле, так что после производства последнего из n продуктов снова возобновляется в том же фиксированном цикле производство первого продукта.

Возникает вопрос о том, может ли быть найдена циклическая последовательность производства продуктов p_i $(i = 1, 2, \ldots, n)$, не требующая перенастройки аппаратуры. Пусть G — ориентированный граф, вершины которого представляют продукты, а существование дуги (x_i, x_j) означает, что продукт p_j может следовать за продуктом p_i без перенастройки аппаратуры. Тогда ответ на поставленный вопрос зависит от того, имеет ли этот орграф гамильтонов цикл [2]) или нет. (Гамильтонов цикл в орграфе — это ориентированный цикл (контур), проходящий ровно один раз через каждую вершину графа; см. гл. 1.)

Если не существует циклической последовательности продуктов, не требующей перенастройки аппаратуры, то какова должна быть последовательность производства с наименьшими затратами на перенастройку, т. е. требующая наименьшего числа необходимых перенастроек? Ответ на этот вопрос может быть получен с помощью итеративного применения алгоритма нахождения гамильтонова цикла в орграфе. Как именно это может быть сделано, обсуждается позже в этой главе.

---

[1]) Сформулированная выше задача может возникать в следующих двух случаях. Или стоимость перенастройки действительно не зависит от продукта, или, другая возможность, стоимость в каждом случае неизвестна и в качестве приближения берется средняя постоянная стоимость.

[2]) Точнее, гамильтонов контур.— *Прим. ред.*

Таким образом, метод, позволяющий ответить на вопрос, содержит ли какой-либо орграф гамильтонов цикл, имеет прямые приложения в задачах упорядочения или планирования операций. В равной степени важно использование такого метода в качестве основного шага в алгоритмах решения других, на первый взгляд далеких от данной тематики, задач теории графов.

В этой главе рассматриваются следующие две задачи.

Задача i. Дан ориентированный граф G, требуется найти в G гамильтонов цикл (или все циклы), если существует хотя бы один такой цикл.

Задача ii. Дан *полный* орграф G, дугам которого приписаны произвольные веса $\mathbf{C} = [c_{ij}]$, найти такой гамильтонов цикл (цепь), который имеет наименьший общий вес. Задача нахождения гамильтонова цикла с наименьшим весом хорошо известна в литературе как *задача коммивояжера* [1, 2, 15]. Следует отметить, что если орграф G не полный, то его можно рассматривать как полный орграф, приписывая отсутствующим дугам бесконечный вес.

Алгоритмы решения задачи коммивояжера и ее вариантов имеют большое число практических приложений в различных областях человеческой деятельности. Рассмотрим, например, задачу, в которой грузовик выезжает с центральной базы для доставки товаров данному числу потребителей и возвращается назад на базу. Стоимость перевозки пропорциональна пройденному грузовиком расстоянию, и при заданной матрице расстояний между потребителями маршрут с наименьшими транспортными затратами получается как решение соответствующей задачи коммивояжера. Аналогичные типы задач возникают при сборе почтовых отправлений из почтовых ящиков, составлении графика движения школьных автобусов по заданным остановкам и т. д. Задача очень легко обобщается и на тот случай, когда доставкой (сбором) занимаются несколько грузовиков, хотя эту задачу можно также переформулировать как задачу коммивояжера бо́льшей размерности [9]. Другие приложения включают составление расписания выполнения операций на машинах [5], проектирование электрических сетей [3], управление автоматическими линиями [17] и т. д.

Вполне очевидно, что сформулированная выше задача (i) является частным случаем задачи (ii). В самом деле, приписывая случайным образом дугам заданного орграфа G конечные веса, получаем задачу коммивояжера [1]). Если решение для этой задачи, т. е. кратчайший гамильтонов цикл [2]), имеет конечное значение, то это решение является гамильтоновым циклом орграфа G (т. е.

[1]) Здесь, как обычно, предполагается, что орграф G «расширен» до некоторого полного орграфа G' и дугам, отсутствующим в G, но принадлежащим полному орграфу G', приписаны бесконечные веса.— *Прим. ред.*

[2]) Так (для краткости) называется гамильтонов цикл с наименьшим общим весом.— *Прим. ред.*

ответом на задачу i). Если же решение имеет бесконечное значение, то G не имеет гамильтонова цикла. С другой стороны можно дать еще одну интерпретацию задачи i). Рассмотрим снова *полный* орграф G_1 с общей матрицей весов дуг $[c_{ij}]$ и рассмотрим задачу нахождения такого гамильтонова цикла, в котором *самая длинная* дуга [1]) минимальна. Эту задачу можно назвать *минимаксной задачей коммивояжера*, оттеняя ее «минимаксную природу» (по сравнению с классической задачей коммивояжера), которую в той же терминологии можно было бы назвать *минисуммной* задачей. Покажем теперь, что задача (i) действительно эквивалентна минимаксной задаче коммивояжера.

В вышеупомянутом полном орграфе G_1 мы можем наверняка найти гамильтонов цикл. Пусть это будет цикл Φ_1, и пусть вес самой длинной его дуги равен $\hat{c}_1$. Удалив из G_1 любую дугу, вес которой не меньше $\hat{c}_1$, получим орграф G_2. Найдем в орграфе G_2 гамильтонов цикл Φ_2, и пусть вес его самой длинной дуги равен $\hat{c}_2$. Удалим из G_2 любую дугу, вес которой не меньше $\hat{c}_2$, и так будем продолжать до тех пор, пока не получим орграф G_{m+1}, не содержащий никакого гамильтонова цикла. Гамильтонов цикл Φ_m в G_m (с весом $\hat{c}_m$) является тогда по определению решением минимаксной задачи коммивояжера, так как из отсутствия гамильтонова цикла в G_{m+1} следует, что в G_1 не существует никакого гамильтонова цикла, не использующего по крайней мере одну дугу с весом, большим или равным $\hat{c}_m$. Таким образом, алгоритм нахождения гамильтонова цикла в орграфе решает также минимаксную задачу коммивояжера. Наоборот, если мы располагаем алгоритмом решения последней задачи, то гамильтонов цикл в произвольном орграфе G может быть найден с помощью построения полного орграфа G_1 с тем же самым множеством вершин, что и в G, дугам которого, соответствующим дугам из G, приписаны единичные веса, а остальным дугам — бесконечные веса. Если решение минимаксной задачи коммивояжера для G_1 имеет конечный вес (на самом деле равный единице), то в графе G может быть найден соответствующий гамильтонов цикл. Если же решение имеет бесконечный вес, то в графе G не существует никакого гамильтонова цикла. Следовательно, две указанные задачи можно рассматривать как эквивалентные, поскольку было продемонстрировано, что алгоритм нахождения гамильтонова цикла позволяет решать минимаксную задачу коммивояжера и наоборот.

Ввиду того что обе сформулированные выше задачи (i) и (ii) часто встречаются в практических ситуациях и (как мы увидим позже) задачу (i) саму по себе решить намного проще, чем как подзадачу задачи (ii), мы обе эти задачи рассмотрим отдельно в частях I и II настоящей главы.

[1]) То есть дуга с наибольшим весом.— *Прим. ред.*

ЧАСТЬ I

2. Гамильтоновы циклы в графе

Пока неизвестно никакого простого критерия или алгебраического метода, позволяющего ответить на вопрос, существует или нет в произвольном графе G гамильтонов цикл. Критерии существования, данные в работах Поша [29], Нэша-Уильямса [25] и Оре [26], представляют теоретический интерес, но являются слишком общими и не пригодны для произвольных графов, встречающихся на практике (см. задачу 10.1). Алгебраические методы определения гамильтоновых циклов, появившиеся в литературе, не могут быть применены к задачам с более чем несколькими десятками вершин, так как они требуют слишком большого времени работы и большой памяти компьютера. Более приемлемым является способ Робертса и Флореса [30, 31], который не предъявляет чрезмерных требований к памяти компьютера, но время в котором зависит экспоненциально от числа вершин в графе. Однако другой неявный метод перебора [35, 6] имеет для большинства типов графов очень небольшой показатель роста времени вычислений в зависимости от числа вершин. Он может быть использован для нахождения гамильтоновых циклов в очень больших графах. В настоящей главе мы опишем алгебраический метод и два способа перебора.

2.1. Алгебраический метод

Этот метод основан на работе Йоу [37], Даниэльсона [11] и Дхавана [14] и включает в себя построение всех простых цепей с помощью последовательного перемножения матриц.

«Внутреннее произведение вершин» цепи $x_1, x_2, x_3, \ldots, x_{k-1}, x_k$ определяется как выражение вида $x_2 \cdot x_3 \cdot \ldots \cdot x_{k-1}$, не содержащее две концевые вершины [1]) x_1 и x_k. «Модифицированная матрица смежности» $\mathbf{B} = [\beta(i, j)]$ — это $(n \times n)$-матрица, в которой $\beta(i, j) = x_j$, если существует дуга из x_i в x_j и нуль в противном случае. Предположим теперь, что у нас есть матрица $\mathbf{P}_l = [p_l(i, j)]$, где $p_l(i, j)$ — сумма внутренних произведений всех простых цепей длины l ($l \geqslant 1$) между вершинами x_i и x_j для $x_i \neq x_j$. Положим $p_l(i, i) = 0$ для всех i. Обычное алгебраическое произведение матриц $\mathbf{B} \cdot \mathbf{P}_l = \mathbf{P}'_{l+1} = [p'_{l+1}(s, t)]$ определяется как

$$p'_{l+1}(s, t) = \sum_k \beta(s, k) \cdot p_l(k, t), \tag{10.1}$$

т. е. $p'_{l+1}(s, t)$ является суммой внутренних произведений всех цепей из x_s в x_t длины $l + 1$. Так как все цепи из x_k в x_t, представленные внутренними произведениями из $p_l(k, t)$, являются

[1]) Часто точки между буквами будут опускаться. При $k = 2$ произведение равно 1.— *Прим. ред.*

простыми, то среди цепей, получающихся из выражения (10.1), не являются простыми лишь те, внутренние произведения которых в $p_l(k, t)$ содержат вершину x_s. Таким образом, если из $p'_{l+1}(s, t)$ исключить все слагаемые, содержащие x_s (а это можно сделать простой проверкой), то получим $p_{l+1}(s, t)$. Матрица $\mathbf{P}_{l+1} = [p_{l+1}(s, t)]$, все диагональные элементы которой равны 0, является тогда матрицей всех простых цепей длины $l + 1$.

Вычисляя затем $\mathbf{B} \cdot \mathbf{P}_{l+1}$, находим $\mathbf{P}_{l+2}$ и т. д., пока не будет построена матрица $\mathbf{P}_{n-1}$, дающая все гамильтоновы цепи (имеющие длину $n - 1$) между всеми парами вершин. Гамильтоновы циклы получаются тогда сразу из цепей в $\mathbf{P}_{n-1}$ и тех дуг из G, которые соединяют начальную и конечную вершины каждой цепи. С другой стороны, гамильтоновы циклы даются членами внутреннего произведения вершин, стоящими в любой диагональной ячейке матрицы $\mathbf{B} \cdot \mathbf{P}_{n-1}$ (все диагональные элементы этой матрицы одинаковые).

Очевидно, что в качестве начального значения матрицы $\mathbf{P}$ (т. е. $\mathbf{P}_1$) следует взять матрицу смежности $\mathbf{A}$ графа, положив все ее диагональные элементы равными нулю.

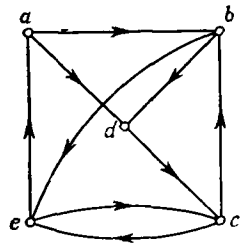


Рис. 10.1. Граф из примера 2.1.1.

2.1.1. Пример. Рассмотрим граф, изображенный на рис. 10.1, матрица смежности которого равна

$$\mathbf{A} = \begin{array}{c|ccccc} & a & b & c & d & e \\ \hline a & 0 & 1 & 0 & 1 & 0 \\ b & 0 & 0 & 0 & 1 & 1 \\ c & 0 & 1 & 0 & 0 & 1 \\ d & 0 & 0 & 1 & 0 & 0 \\ e & 1 & 0 & 1 & 0 & 0 \end{array}$$

и модифицированная матрица смежности

$$\mathbf{B} = \begin{array}{c|ccccc} & a & b & c & d & e \\ \hline a & 0 & b & 0 & d & 0 \\ b & 0 & 0 & 0 & d & e \\ c & 0 & b & 0 & 0 & e \\ d & 0 & 0 & c & 0 & 0 \\ e & a & 0 & c & 0 & 0 \end{array}$$

Положим $\mathbf{P}_1 \equiv \mathbf{A}$. Матрица $\mathbf{P}_2' = \mathbf{B} \cdot \mathbf{P}_1'$ получается равной

$$\mathbf{P}_2' = \begin{array}{c|ccccc} & a & b & c & d & e \\ \hline a & 0 & 0 & d & b & b \\ b & e & 0 & d+e & 0 & 0 \\ c & e & 0 & \underline{e} & b & b \\ d & 0 & c & 0 & 0 & c \\ e & 0 & a+c & 0 & a & \underline{c} \end{array}$$

Матрица $\mathbf{P}_2$ почти такая же, как $\mathbf{P}_2'$, — только подчеркнутые элементы в $\mathbf{P}_2'$ надо заменить нулями. Матрица $\mathbf{P}_3 = \mathbf{B} \cdot \mathbf{P}_2$ равна

$$\mathbf{P}_3' = \begin{array}{c|ccccc} & a & b & c & d & e \\ \hline a & \underline{be} & dc & bd+be & 0 & dc \\ b & 0 & \underline{dc+ea}+ec & 0 & ea & dc \\ c & be & ea+\underline{ec} & \underline{bd+be} & ea & 0 \\ d & ce & 0 & 0 & \underline{cb} & cb \\ e & \underline{ce} & 0 & ad & ab+cb & \underline{ab+cb} \end{array}$$

$\mathbf{P}_3$ получается из $\mathbf{P}'_3$ после замены подчеркнутых элементов нулями:

$\mathbf{P}'_4 =$

	a	b	c	d	e
a	$\underline{dce}$	0	0	$\underline{bea}$	$bdc + bdc$
b	dce	0	ead	$\underline{eab + ecb}$	$\underline{dcb}$
c	0	0	$\underline{ead}$	$bea + eab + ecb$	$\underline{bdc}$
d	cbe	cea	0	$\underline{cea}$	0
e	$\underline{cbe}$	$adc + \underline{cea}$	$abd + abe$	$\underline{cea}$	$\underline{adc}$

Матрица $\mathbf{P}_4$ гамильтоновых цепей равна

$\mathbf{P}_4 =$

	a	b	c	d	e
a	0	0	0	0	$bdc + dcb$
b	dce	0	ead	0	0
c	0	0	0	$bea + eab$	0
d	cbe	cea	0	0	0
e	0	adc	abd	0	0

Гамильтоновы цепи $abdce$ и $adcbe$, соответствующие элементу (1,4) вышенаписанной матрицы, дают гамильтоновы циклы $abdcea$ и $adcbea$, если добавить замыкающую дугу (e, a). Все другие гамильтоновы цепи в $\mathbf{P}_4$ приводят к тем же самым двум гамильтоновым циклам, и поэтому в графе G существует только два таких цикла.

Недостатки этого метода совершенно очевидны. В процессе умножения матриц (т. е. когда l увеличивается) каждый элемент матрицы $\mathbf{P}_l$ будет состоять из все большего числа членов вплоть до некоторого критического значения l, после которого число членов снова начнет уменьшаться. Это происходит вследствие того, что для малых значений l и для графов, обычно встречающихся на практике, число цепей длины $l + 1$, как правило, больше, чем число цепей длины l, а для больших значений l имеет место обратная картина. Кроме того, так как длина каждого члена внутреннего произведения вершин увеличивается на единицу, когда l увеличивается на единицу, то объем памяти, необходимый для хранения матрицы $\mathbf{P}_l$, растет очень быстро вплоть до максимума при некотором критическом значении l, после которого этот объем снова начинает уменьшаться.

Небольшая модификация вышеприведенного метода позволяет во много раз уменьшить необходимый объем памяти и время вычислений. Так как нас интересуют только гамильтоновы циклы и, как было отмечено выше, они могут быть получены из членов внутреннего произведения любой диагональной ячейки матрицы $\mathbf{B} \cdot \mathbf{P}_{n-1}$, то необходимо знать только элемент p_{n-1} (1,1). При этом на каждом этапе не обязательно вычислять и хранить всю матрицу $\mathbf{P}_l$, достаточно лишь найти первый столбец из $\mathbf{P}_l$. Эта модификация уменьшает необходимый объем памяти и время вычислений в n раз. Однако даже при использовании этой модификации программа для ЭВМ [14], написанная на языке PL/1, который позволяет построковую обработку литер и переменное распределение памяти, не была способна найти все гамильтоновы циклы в неориентированных графах с более чем примерно 20 вершинами и средним значением степени вершины, большим 4. Использовалась вычислительная машина ИБМ 360/65 с памятью 120 000 байтов. Более того, даже для графа с вышеуказанными «размерами» данный метод использовал фактически весь объем памяти и время вычислений равнялось 1,8 минуты. Не такое уж незначительное время для столь небольшого графа.

2.2. Метод перебора Робертса и Флореса

В противоположность алгебраическим методам, с помощью которых пытаются найти сразу все гамильтоновы циклы и при реализации которых приходится хранить поэтому все цепи, которые могут оказаться частями таких циклов, метод перебора имеет дело с одной цепью, непрерывно продлеваемой вплоть до момента, когда либо получается гамильтонов цикл, либо становится ясно, что эта цепь не может привести к гамильтонову циклу. Тогда цепь модифицируется некоторым систематическим способом (который гарантирует, что в конце концов будут исчерпаны все возможности), после чего продолжается поиск гамильтонова цикла. В этом способе для поиска требуется очень небольшой объем памяти и за один раз находится один гамильтонов цикл.

Следующая схема перебора, использующая обычную технику возвращения, была первоначально предложена Робертсом и Флоресом [30, 31]. Начинают с построения $(k \times n)$-матрицы $\mathbf{M} = [m_{ij}]$, где элемент m_{ij} есть i-я вершина (скажем x_q), для которой в графе $G = (X, \Gamma)$ существует дуга (x_j, x_q). Вершины x_q в множестве $\Gamma(x_j)$ можно упорядочить произвольно, образовав элементы j-го столбца матрицы $\mathbf{M}$. Число строк k матрицы $\mathbf{M}$ будет равно наибольшей полустепени исхода вершины.

Метод состоит в следующем. Некоторая начальная вершина (скажем, x_1) выбирается в качестве отправной и образует первый

элемент множества S, которое каждый раз будет хранить уже найденные вершины строящейся цепи. К S добавляется первая вершина (например, вершина a) в столбце x_1. Затем к множеству S добавляется первая возможная вершина (например, вершина b) в столбце a, потом добавляется к S первая возможная вершина (например, вершина c) в столбце b и т. д. Под «возможной» вершиной мы понимаем вершину, еще не принадлежащую S. Существуют две причины, препятствующие включению некоторой вершины на шаге r в множестве $S = \{x_1, a, b, c, \ldots, x_{r-1}, x_r\}$. Или (1) в столбце x_r нет возможной вершины, или (2) цепь, определяемая последовательностью вершин в S, имеет длину $n-1$, т. е. является гамильтоновой цепью.

В случае (2)

(а) в графе G существует дуга (x_r, x_1) и поэтому найден гамильтонов цикл, или

(б) дуга (x_r, x_1) не существует и не может быть получен никакой гамильтонов цикл.

В случаях (1) и (2б) следует прибегнуть к *возвращению*, в то время как в случае (2а) можно прекратить поиск и напечатать результат (если требуется найти только один гамильтонов цикл), или (если нужны все такие циклы) произвести печать и прибегнуть к возвращению. Возвращение состоит в удалении последней включенной вершины x_r из S, после чего остается множество $S = \{x_1, a, b, c, \ldots, x_{r-1}\}$, и добавлении к S первой возможной вершины, следующей за x_r, в столбце x_{r-1} матрицы **M**. Если не существует никакой возможной вершины, делается следующий шаг возвращения и т. д.

Поиск заканчивается в том случае, когда множество S состоит только из вершины x_1 и не существует никакой возможной вершины, которую можно добавить к S, так что шаг возвращения делает множество S пустым. Гамильтоновы циклы, найденные к этому моменту, являются тогда всеми гамильтоновыми циклами, существующими в графе.

2.2.1. Пример. Рассмотрим граф, изображенный на рис. 10.2. Матрица **M** приводится ниже, вершины в каждом столбце расположены в алфавитном порядке:

$$\mathbf{M} = \begin{array}{c|cccccc} & a & b & c & d & e & f \\ \hline 1 & b & c & a & c & c & a \\ 2 & — & e & d & f & d & b \\ 3 & — & — & — & — & — & — \end{array}$$

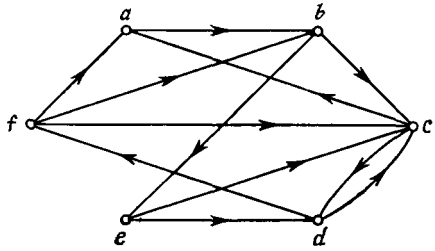


Рис. 10.2. Граф из примера 2.2.1.

Поиск всех гамильтоновых циклов производится так (вершина a берется в качестве отправной вершины):

	Множество S	*Комментарии*
1.	a	Добавляем первую возможную вершину в столбце a (т. е. вершину b).
←2.	a, b	Добавляем первую возможную вершину в столбце b (т. е. вершину c).
3.	a, b, c	Первая вершина (a) в столбце c не является возможной ($a \in S$), добавляем следующую вершину в столбце (т. е. вершину d).
4.	a, b, c, d	Добавляем вершину f.
5.	a, b, c, d, f	В столбце f нет возможной вершины, возвращение.
6.	a, b, c, d	В столбце d не существует возможной вершины, следующей за f. Возвращение.
7.	a, b, c	Аналогично предыдущему. Возвращение.
8.	a, b	Добавляем вершину e.
→9.	a, b, e	Добавляем вершину c.
10.	a, b, e, c	Добавляем вершину d.
11.	a, b, e, c, d	Добавляем вершину f.
12.	a, b, e, c, d, f	Гамильтонова цепь. Дуга (f, a) дает гамильтонов цикл. Возвращение.
13.	a, b, e, c, d	Возвращение.
14.	a, b, e, c	Возвращение.
15.	a, b, e	Добавляем вершину d.
16.	a, b, e, d	Добавляем вершину f.
17.	a, b, e, d, f	Добавляем вершину c.

18.	a, b, e, d, f, c	Гамильтонова цепь. Цепь замыкается дугой (c, a). Возвращение.
19.	a, b, e, d, f	Возвращение.
20.	a, b, e, d	Возвращение.
21.	a, b, e	Возвращение.
22.	$a, b,$	Возвращение.
23.	a	Возвращение.
24.	$\varnothing$	Конец поиска.

2.2.2. Улучшение основного метода. Допустим, что на некотором этапе поиска построенная цепь задается множеством $S = \{x_1, x_2, \ldots, x_r\}$ и что следующей вершиной, которую предполагается добавить к S, является $x^* \notin S$. Рассмотрим теперь две

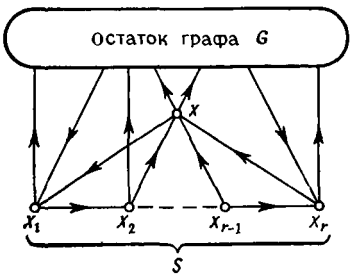


Рис. 10.3а. ($x \in \Gamma(x_r)$ и $\Gamma^{-1}(x) \subset S$.) Следующей дугой должна быть (x_r, x).

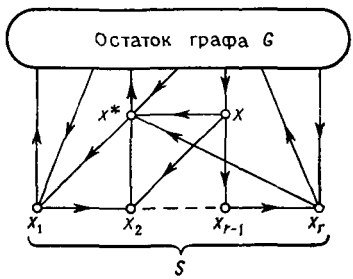


Рис. 10.3б. ($x \notin \Gamma^{-1}(x_1)$ и $\Gamma(x) \subset S \cup \{x^*\}$.) Следующей дугой не должна быть (x_r, x^*).

следующие ситуации, в которых вершина является изолированной в подграфе, остающемся после удаления из $G = (X, \Gamma)$ всех вершин, образующих построенную до этого цепь.

(а) Если существует такая вершина $x \in X - S$, что $x \in \Gamma(x_r)$ и $\Gamma^{-1}(x) \subseteq S$ (см. рис. 10.3а), то, добавляя к S любую вершину x^*, отличную от x, мы не сможем в последующем достигнуть вершины x ни из какой конечной вершины построенной цепи, и, значит, эта цепь не сможет привести нас к построению гамильтонова цикла. Таким образом, в этом случае x является единственной вершиной, которую можно добавить к S для продолжения цепи.

(б) Если существует такая вершина $x \in X - S$, что $x \notin \Gamma^{-1}(x_1)$ и $\Gamma(x) \subset S \cup \{x^*\}$ для некоторой другой вершины x^*, то x^* не может быть добавлена к S, так как тогда в остающемся подграфе не может существовать никакой цепи между x и x_1. Цепь, определяемая множеством $S \cup \{x^*\}$, не может поэтому привести к гамильтонову циклу, а в качестве кандидата на добавление к множеству S следует рассмотреть другую вершину, отличную от x^* (см. рис. 10.3б).

В только что приведенном примере ситуация (а) возникает на шаге 2, когда множество S есть $\{a, b\}$. Если заметить теперь, что для вершины e справедливо соотношение $\Gamma^{-1}(e) = \{b\} \subset S$, то становится ясным, что e следует добавить в качестве очередной вершины к множеству $\{a, b\}$. Поэтому в приведенном примере можно опустить шаги 3—8 и сразу от шага 2 перейти, как показано стрелкой, к шагу 9.

Проверка условий (а) и (б) будет, конечно, замедлять итеративную процедуру, и для небольших графов (менее чем с 20 вершинами) не получается никакого улучшения первоначального алгоритма Робертса и Флореса. Но для больших графов эта проверка приводит к заметному сокращению необходимого времени вычислений, уменьшая его обычно в 2 или более раз. Подробные результаты вычислений для различных методов приводятся на рис. 10.7 в разд. 3.

2.3. Мультицепной метод

После внимательного изучения операций алгоритма перебора Робертса и Флореса становится очевидным, что даже после сделанного улучшения не слишком много внимания уделяется оставшейся части графа, в которой берется последовательность вершин, продолжающих построенную цепь. Обычно построение цепи S_0 в процессе поиска (S_0 рассматривается и как упорядоченное множество вершин, и как обычное множество) подразумевает существование еще каких-то цепей в других частях графа. Эти предполагаемые цепи либо помогают быстрее построить гамильтонов цикл, либо указывают на отсутствие такого цикла, содержащего цепь S_0, что позволяет сразу прибегнуть к возвращению.

Метод, описанный в этом разделе, был предложен первоначально Селби [35] для неориентированных графов; здесь дается его небольшое видоизменение для ориентированных графов. Метод состоит в следующем.

Допустим, что на некотором этапе поиска построена цепь S_0 и возможны цепи $S_1, S_2, \ldots$. Рассмотрим какую-либо «среднюю» вершину одной из этих цепей (слово «средняя» здесь означает любую вершину, отличную от начальной и конечной). Поскольку эта вершина уже включена в цепь с помощью двух дуг, то очевидно, что все другие дуги, входящие или выходящие из такой вершины, могут быть удалены из графа. Для любой начальной вершины вышеуказанных цепей можно удалить все дуги, исходящие из нее (за исключением дуги, включающей эту вершину в цепь), а для любой конечной вершины можно удалить все дуги, оканчивающиеся в ней (опять-таки за исключением дуги, включающей ее в цепь). Кроме того, за исключением слу-

чая, когда существует только одна цепь (скажем, S_0), проходящая через все вершины графа G (т. е. когда S_0 — гамильтонова цепь), любая имеющаяся дуга, ведущая из конца любой цепи в начальную вершину этой же цепи, может быть удалена, так как такая дуга замыкает не гамильтоновы циклы.

Удаление всех этих дуг даст граф со многими вершинами — всеми «средними» вершинами цепей — в котором только одна дуга оканчивается в каждой вершине и только одна дуга исходит из нее. Все эти «средние» вершины и дуги, инцидентные им, удаляются из G, а вместо них для каждой цепи вводится единственная дуга, идущая от начальной вершины цепи до ее конечной вершины. В результате всего этого получается *редуцированный граф* $G_k = (X_k, \Gamma_k)$, где k — индекс, показывающий номер шага поиска.

Рассмотрим теперь продолжение цепи S_0 (сформированной в результате поиска), осуществляемое путем добавления вершины x_j, которая является возможной в смысле алгоритма Робертса и Флореса, т. е. в G_k существует дуга, исходящая из конечной вершины цепи S_0 — обозначим эту вершину $e(S_0)$ — и входящая в вершину x_j. Добавление x_j к S_0 осуществляется так:

(1) Сначала удаляются из G_k все необходимые дуги, т. е.

(а) все дуги, оканчивающиеся в x_j или исходящие из $e(S_0)$, за исключением дуги $(e(S_0), x_j)$;

(б) все дуги, выходящие из x_j в начальную вершину пути S_0;

(в) если окажется, что x_j является начальной вершиной другой цепи S_j, то следует удалить также любую дугу, ведущую из конечной вершины цепи S_j в начальную вершину цепи S_0.

(2) Обозначим граф, оставшийся после удаления всех дуг, через $G'_k = (X_k, \Gamma_{k'})$.

Если существует вершина x в графе G'_k, не являющаяся конечной ни для одной из цепей $S_0, S_1, \ldots$ и которая после удаления дуг имеет полустепень захода, равную единице, т. е. $|\Gamma_k^{-1}(x)| = 1$, то выкинуть все дуги, исходящие из вершины $v = \Gamma_k^{-1}(x)$, за исключением дуги (v, x).

Если существует вершина x графа G'_k, не являющаяся начальной ни для какой цепи и которая после удаления дуг имеет полустепень исхода, равную единице, т. е. $|\Gamma_{k'}(x)| = 1$, то выкинуть все дуги, исходящие из вершины x, за исключением дуги $(x, \Gamma_{k'}(x))$.

Перестроить все цепи и удалить дуги, ведущие из конечных в начальные вершины.

Повторять шаг 2 до тех пор, пока можно удалять дуги.

(3) Удалить из оставшегося графа G'_k все вершины, полустепени захода и исхода которых равны единице, т. е. вершины, которые стали теперь «средними» вершинами цепей. Это удаление производится так, как это было описано выше, в результате чего

получается новый редуцированный граф G_{k+1}, заменяющий предыдущий граф G_k.

Совершенно очевидно, что если добавление вершины x_j к цепи S_o делает полустепень захода или полустепень исхода (или обе) некоторой вершины x в конце шага 2 равной нулю, то не существует никакого гамильтонова цикла. В этом случае вершина x_j удаляется из множества S_o и в качестве другой вершины x_j, позволяющей продолжить цепь S_o, выбирается некоторая другая вершина из множества $\Gamma_{k'}[e(S_o)]$, и так до тех пор, пока не будет исчерпано все множество $\Gamma_{k'}[e(S_o)]$ и придется прибегнуть к возвращению (т. е. $e(S_o)$ удаляется из S_o и заменяется другой вершиной и т. д.). Отметим, что операция возвращения предполагает хранение достаточной информации об удаленных дугах в шагах 1 и 2 на каждом этапе k, чтобы можно было по графу G_{k+1} восстановить граф G_k при любых k, если приходится прибегать к возвращению.

Если (на некотором этапе) в конце шага 2 установлено, что только одна цепь проходит далее через все вершины, то в этом случае существование гамильтонова цикла может быть выявлено непосредственно. Если цикл при этом не найден (или если он найден, но надо найти все гамильтоновы циклы), то нужно прибегнуть к возвращению.

Лучший вычислительный путь состоит в том, чтобы при каждой итерации шага 2 (за исключением последней) проверять отличие от нуля полустепеней захода и исхода всех вершин графа G'_k. Поэтому как только одна из них станет равной нулю, сразу же применяется операция возвращения, и если найдена гамильтонова цепь, то получается и гамильтонов цикл без проверки существования дуги возврата [1]).

Если не возникает ни один из вышеупомянутых случаев, т. е. если (на некотором этапе k) в конце шага 2 остается более чем одна цепь и все полустепени являются ненулевыми, то нельзя еще сделать никаких выводов. Тогда вершина x_j добавляется к S_o и выбирается другая вершина для дальнейшего продолжения цепи. Шаги 1, 2, 3 повторяются, начиная с нового редуцированного графа.

2.3.1. Пример. В настоящем примере будет показано, как итеративный процесс шага 2 приводит ко многим цепям, получающимся из основной цепи S_o, и как эти цепи могут привести к быстрому окончанию производимого поиска.

Рассмотрим часть графа, изображенную на рис. 10.4. Сначала полустепени захода и исхода каждой вершины больше единицы.

---

[1]) Здесь дуга, замыкающая гамильтонову цепь.— *Прим. ред.*

Первая итерация. Начнем с вершины 1 как начальной вершины строящейся цепи S_o, и попробуем добавить к S_o, например, вершину 6, так что цепь S_o будет {1, 6}.

Шаг 1 описанного метода удалит тогда из G дуги (1,2), (1,3), (1,10), (2,6).

Шаг 2 укажет теперь вершину 10, как имеющую полустепень захода 1 и дающую возможную цепь $S_1 = \{4,10\}$, после чего

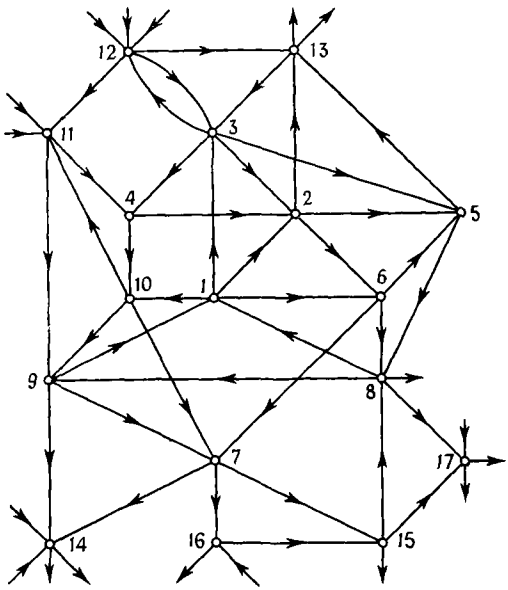


Рис. 10.4. Граф из примера разд. 2.3.1.

удаляется дуга (4,2). Второе применение шага 2 дает вершину 2 с полустепенью захода 1 и возможную цепь $S_2 = \{3,2\}$, после чего удаляются дуги (3,12), (3,5) и (3,4). Третье применение шага 2 даст вершину 4 с полустепенью захода 1 и приводит к продолжению цепи S_1 после добавления дуги (11,4), так что цепь S_1 есть теперь {11,4,10}. В силу шага 2 дуга (11,9) удаляется, и то же относится к дуге (10,11), замыкающей цепь S_1. Четвертое применение шага 2 не приводит к дальнейшим удалениям, и первый редуцированный граф, получающийся в конце шага 3, показан на рис. 10.5а, где цепи S_o, S_1 и S_2 изображены жирными линиями.

Вторая итерация. Так как существует более чем одна цепь и так как полустепени захода и исхода всех вершин ненулевые,

то мы продолжаем расширять цепь S_0 дальше. Пусть, например, из множества $\Gamma(e(S_0)) = \{8,5,7\}$ выбрана вершина 8, так что цепь S_0 превратится в цепь $\{1, 6, 8\}$. Шаг 1 даст теперь вершину 5, обе полустепени которой равны 1, и вершину 1 с полустепенью захода, равной 1. Взяв сначала вершину 5 и сделав два раза шаг

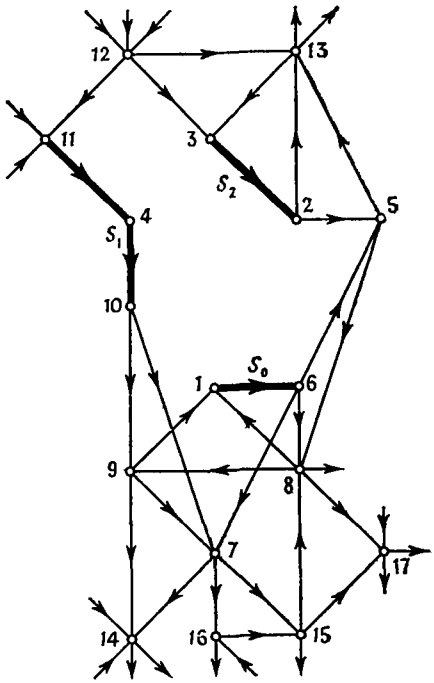


Рис. 10.5а. Первый редуцированный граф G_1.

2, придем к добавлению дуг (2,5) и (5,13) для расширения цепи S_2, которая превращается теперь в $\{3, 2, 5, 13\}$. Дуги (2, 13), (12, 13) и дуга возврата (13, 3) удаляются [1]). Второе применение шага 2 дает вершину 3 с полустепенью захода 1, а также вершину 1 (с полустепенью захода 1), полученную на предыдущем этапе. Взяв вершину 3, добавим к S_2 дугу (12, 3), после чего получим цепь $\{12, 3, 2, 5, 13\}$ и удалим дугу (12, 11). Третье применение шага 2 все еще дает вершину 1 с полустепенью захода 1 и больше не дает никаких вершин. К цепи S_0 добавляется дуга (9,1), цепь S_0 превращается в $\{9, 1, 6, 8\}$, а дуги (9, 7), (9,14) и дуга возвра-

[1]) Здесь термин «дуга возврата» используется в несколько ином смысле, чем на стр. 255: эта дуга замыкает не гамильтонову, а некоторую промежуточную цепь.— *Прим. ред.*

та (8,9) удаляются. Четвертое применение шага 2 укажет две вершины 9 и 7 с полустепенями захода 1. Взяв сначала вершину 9, добавим дугу (10,9), соединяющую цепи S_1 и S_0 и дающую новую цепь $S_0 = \{11, 4, 10, 9, 1, 6, 8\}$. Дуга (10, 7) удаляется.

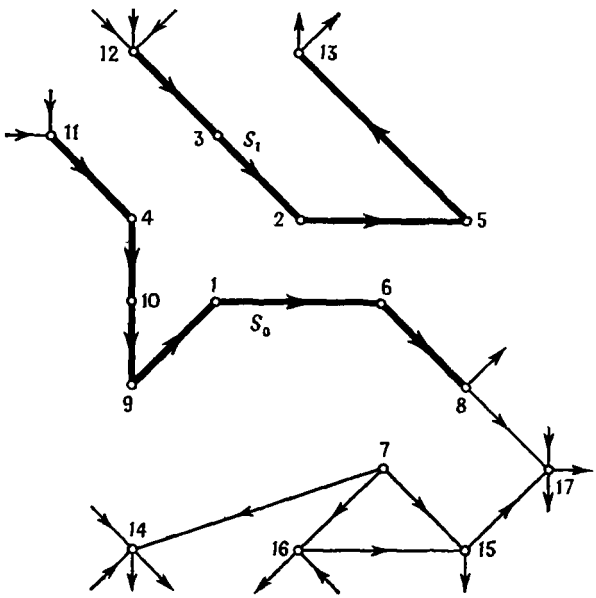


Рис. 10.5б. Граф G_1' после четвертого применения шага 2 на второй итерации.

Пятое применение шага 2 дает вершину 7 с полустепенью захода 0. Это говорит о том, что описанный поиск не может привести ни к какому гамильтонову циклу.

Граф G_1', получающийся в конце четвертого применения шага 2, показан на рис. 10.5б. Теперь следует восстановить все удаленные во время второй итерации дуги, чтобы вернуться к графу G_1 из рис. 10.5а, и произвести операцию возвращения, т. е. удалить из S_0 вершину 8, заменить ее другой возможной вершиной (5 или 7) и продолжать применять шаги 1, 2, 3 и т. д.

Из приведенного примера видно, что это очень сильный метод поиска. После двух итераций он позволяет сделать вывод, что никакой гамильтонов цикл не может содержать в качестве своей части цепь $\{1, 6, 8\}$. Это 1/12 всех попыток поиска, поскольку полустепень исхода вершины 1 равна 4, а вершины 6 — 3. Конечно, указанный вывод относится только к части графа, изображенной на рис. 10.4, которая сама может принадлежать к намного большему графу. Можно поэтому ожидать, что при заданном

среднем значении степеней вершин поиск лишь слабо зависит от размера (числа вершин) графа. Этот факт будет продемонстрирован на экспериментальных данных в следующем разделе.

Напротив, метод Робертса и Флореса (или его улучшенный вариант) для достижения того же результата требует очень длительного поиска, так как в нем существенно перебираются все цепи, начинающиеся с 1, 6, 8, . . . , прежде чем произойдет возвращение к 1, 6,. . . . Очевидно, что затрачиваемая работа зависит как от размера графа, так и от степеней вершин.

3. Сравнение методов поиска гамильтоновых циклов

В настоящем разделе сравниваются первоначальный вариант алгоритма Робертса и Флореса, его улучшенный вариант и мультицепной метод. Эти три метода сравниваются по необходимому времени вычисления для нахождения одного гамильтонова цикла, если таковой существует. или доказательства его отсутствия. Проверка была проведена на случайно выбранных графах, степени вершин которых лежат в предписанных границах. Всего было использовано около 200 графов, для которых приводятся средние результаты. Во всех графах оказались гамильтоновы циклы.

На рис. 10.6 показана зависимость требуемого алгоритмом Робертса и Флореса времени вычисления от числа вершин графа; степени вершин лежат в пределах 3 — 5. Ввиду сильных вариаций требуемого времени для графов одинаковых размеров приводятся три кривые, характеризующие среднее, максимальное и минимальное время, полученное для различных графов с одинаковым числом вершин. Следует заметить, что на рис. 10.6 применен полулогарифмический масштаб, что говорит об экспоненциальном характере зависимости. Формула, дающая приближенную зависимость времени T от числа вершин n графа со степенями вершин в пределах 3 — 5, такова:

$$T = 0{,}85 \cdot 10^{-4} \cdot 10^{0{,}155n} \text{ (секунд на CDC 6600).}$$

Улучшенный вариант алгоритма Робертса и Флореса не намного лучше первоначального алгоритма. Необходимое время вычисления в нем все еще зависит (более или менее) экспоненциально от n. Зависимость отношения времен вычисления при использовании этих двух алгоритмов для неориентированных графов со степенями вершин 3 — 5 приведена на рис. 10.7. Из этого рисунка видно, что «улучшенный» вариант действительно хуже для графов малых размеров, хотя для больших графов (с более чем 20 вершинами) он позволяет сэкономить более 50% времени вычисления.

По отношению к тому же вышеупомянутому множеству графов мультицепной алгоритм оказался очень эффективным. Это видно из рис. 10.8, на котором показано необходимое для этого алгоритма время вычисления (здесь применен линейный масштаб). График показывает, что время растет *очень медленно* в зависимости

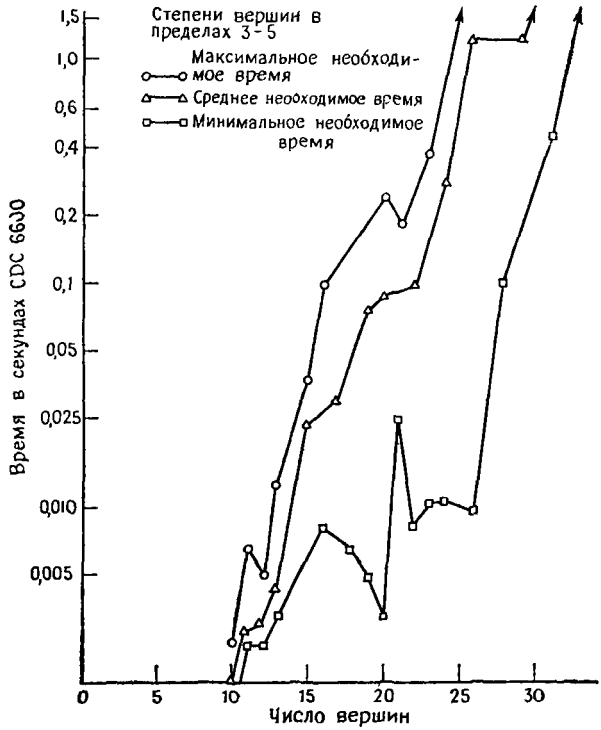


Рис. 10.6. Вычислительная реализация алгоритма Робертса и Флореса.

от числа вершин и поэтому алгоритм применим для очень больших графов [1]). Другим преимуществом этого метода является очень слабая вариация времени вычисления для различных графов одинакового размера, и поэтому можно оценить с разумной степенью доверительности время вычисления, необходимое для различных задач. Кроме того, эксперименты показывают, что

[1]) Отсюда не следует делать вывод, что алгоритм *гарантированно* заканчивается за время, пропорциональное n^k при некотором $k > 0$. На самом деле можно подобрать примеры, которые требуют намного большего времени, чем это показано на рис. 10.8. Такие контрпримеры строятся с помощью графов, не содержащих гамильтоновых циклов.

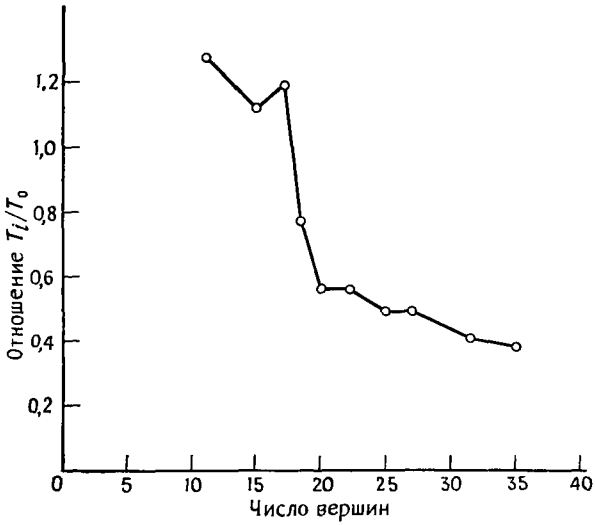


Рис. 10.7. Реализация улучшенного алгоритма Робертса и Флореса. T_0 — время вычисления для первоначального метода, T_i — время вычисления для улучшенного метода.

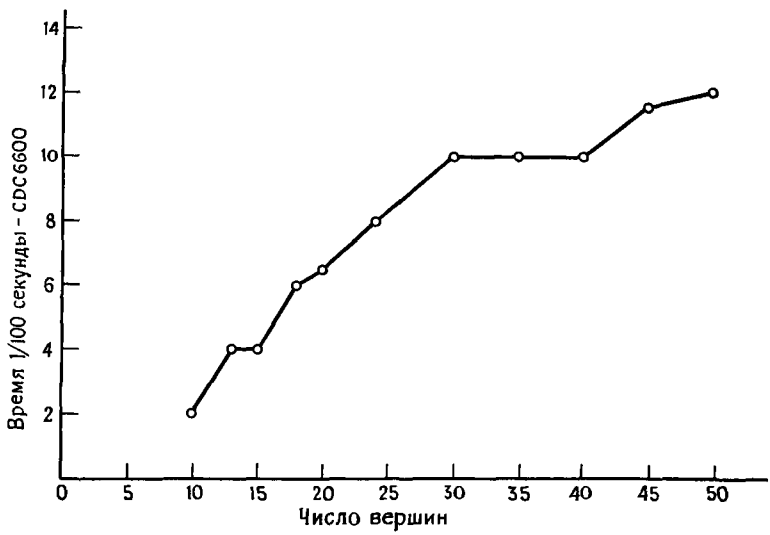


Рис. 10.8. Вычислительная реализация мультицепного алгоритма.

для графов, степени вершин которых лежат в вышеприведенных пределах 3 — 5, метод по существу не чувствителен к степеням вершин.

Вычислительные результаты, показанные на рис. 10.6 — 10.8, относятся к поиску одного гамильтонова цикла в графе. Небезынтересно сказать несколько слов о вычислениях с тремя алгоритмами, когда искались все гамильтоновы циклы. Так, для неориентированного графа с 20 вершинами со степенями вершин 3 —5 потребовалось 2 с, чтобы найти все гамильтоновы циклы, следуя алгоритму Робертса и Флореса (этих циклов оказалось 18). Улучшенный вариант того же алгоритма потребовал 1,2 с, а мультицепной алгоритм — 0,07 с. Вычисления проводились на ЭВМ CDC 6600.

4. Простая задача планирования

Во введении к данной главе мы описали пример промышленной фирмы, выпускающей продукты $p_1, p_2, \ldots, p_n$ с использованием единственного типа аппаратуры в циклическом режиме, и поставили вопрос о порядке производства продуктов, требующем наименьшего возможного числа перенастроек аппаратуры. Был построен граф $G = (X, A)$, вершины x_i которого представляют продукты p_i $(i = 1, 2, \ldots, n)$, а дуги (x_i, x_j) показывают, что продукт p_j можно производить вслед за продуктом p_i без перенастройки.

Если граф $G = (X, A)$ имеет гамильтонов цикл, например $x_{i_1}, x_{i_2}, x_{i_3}, \ldots, x_{i_n}$, то соответствующая последовательность продуктов $p_{i_1}, p_{i_2}, p_{i_3}, \ldots, p_{i_n}$ может быть произведена на аппаратуре без какой-либо перенастройки, так как по определению гамильтонова цикла
$(x_{i_k}, x_{i_{k+1}}) \in A$ для всех $k = 1, 2, \ldots, (x_{i_{n+1}} \equiv x_{i_1})$ и A определено выше как множество $\{(x_i, x_j) \mid c_{ij} = 0\}$.

Если G не имеет никакого гамильтонова цикла, то мы можем построить граф $G_1 = (X_1, A_1)$, где

$$X_1 = X \cup \{y_1\}$$

и

$$A_1 = A \cup \{(x, y_1) \mid x \in X\} \cup \{(y_1, x) \mid x \in X\},$$

т. е. ввести в граф G фиктивную вершину y_1 вместе с дугами, ведущими в нее и исходящими из нее в каждую действительную вершину графа G.

Если у графа G_1 существует гамильтонов цикл, то он будет иметь вид $x_{i_1}, x_{i_2}, \ldots, x_{i_r}, \boxed{y_1}, x_{i_{r+1}}, \ldots, x_{i_n}$, что приводит к сле-

дующей последовательности производства продуктов

$$p_{i_{r+1}},\ p_{i_{r+2}},\ \ldots,\ p_{i_n},\ p_{i_1},\ p_{i_2},\ \ldots,\ p_{i_r}$$

с единственной операцией перенастройки после окончания производства последнего продукта и началом производства первого. Таким образом, фиктивная вершина играет роль метки, показывающей то место в последовательности, где необходима перенастройка аппаратуры. В терминах графа фиктивная вершина и ассоциированные с ней дуги обеспечивают существование цепи между двумя действительными вершинами. Таким образом, если существует последовательность продуктов с одной перенастройкой, т. е. если в G существует гамильтонов цикл, при условии, что можно использовать одну «дугу» $(x_i, x_j) \notin A$, то добавление вершины y_1 к G всегда приведет к существованию в G_1 гамильтонова цикла, так как лишняя необходимая «дуга» (x_i, x_j) может быть заменена двумя фиктивными дугами (x_i, y_1) и (y_1, x_j).

Если граф G_1 не имеет гамильтонова цикла, то мы построим граф $G_2 = (X_2, A_2)$ вместо графа G_1, где

$$X_2 = X_1 \cup \{y_2\}$$

и

$$A_2 = A_1 \cup \{(x, y_2) \mid x \in X\} \cup \{(y_2, x) \mid x \in X\},$$

и аналогично будем действовать далее.

Теорема 1. *Если граф* $G_m = (X_m, A_m)$, *определяемый как*

$$X_m = X \cup [\bigcup_{j=1}^{m} \{y_j\}], \tag{10.2}$$

$$A_m = A \cup [\bigcup_{j=1}^{m} \{(x, y_j) \mid x \in X\}] \cup [\sum_{j=1}^{m} \{(y_j, x) \mid x \in X\}], \tag{10.3}$$

содержит гамильтонов цикл, а граф G_{m-1} *такого цикла не содержит, то число* m *есть минимальное число необходимых перенастроек, и если гамильтонов цикл в* G_m *имеет вид*

$$x_{i_1},\ \ldots,\ x_{i_\alpha},\ \boxed{y_1},\ x_{i_{\alpha+1}},\ \ldots,\ x_{i_\beta},\ \boxed{y_2},\ x_{i_{\beta+1}},\ \ldots,\ x_{i_\gamma},\ \boxed{y_3},$$

$$x_{i_{\gamma+1}},\ \ldots,\ x_{i_\delta},\ \boxed{y_m},\ x_{i_{\delta+1}},\ \ldots,\ x_{i_n},$$

то продукты должны производиться в последовательности

$$p_{i_{\alpha+1}},\ \ldots,\ p_{i_\beta},\ \textit{затем}\ p_{i_{\beta+1}},\ \ldots,\ p_{i_\gamma},\ \ldots\ \textit{и т. д.}\ \ldots$$

$$\textit{затем}\ p_{i_{\delta+1}},\ \ldots,\ p_{i_n}, p_{i_1},\ \ldots,\ p_{i_\alpha}.$$

Доказательство. Доказательство следует непосредственно из рассуждений, предшествующих теореме, с использованием индукции.

4.1. Вычислительные аспекты

В рассуждениях предыдущего раздела граф G каждый раз пополнялся единственной фиктивной вершиной. Если оптимальное решение задачи содержит m перенастроек аппаратуры, то нужно будет сделать $m + 1$ попыток отыскать гамильтоновы циклы в графах C, G_1, . . ., G_m, причем только последняя из этих попыток окажется успешной и приведет к решению задачи. Очевидно, что число m ограничено сверху величиной n и в общем случае в практических задачах оно будет лишь небольшой частью числа n. Тем не менее с вычислительной точки зрения необходимы различные процедуры для проверки наличия в графе гамильтоновых циклов и построения таких циклов, так как оказывается [6], что проще найти гамильтонов цикл в графе, имеющем такой цикл, чем доказать, что не существует никакого гамильтонова цикла в графе, его не имеющем. Этот факт подсказывает, что более подходящим будет алгоритм, который начинает с верхней границы B для оптимального (минимального) числа перенастроек аппаратуры и последовательно дает и проверяет графы G_B, G_{B-1}, . . . и т. д. до тех пор, пока не будет найден граф G_{m-1}, не имеющий гамильтоновых циклов. Краткое описание такого алгоритма дается ниже.

Шаг 1. Найти верхнюю границу B для оптимального (минимального) числа необходимых перенастроек аппаратуры (см. приложение).

Шаг 2. Используя формулы (10.2) и (10.3), построить граф G_B.

Шаг 3. Определить, имеет ли граф G_B гамильтонов цикл. Если да, то хранить этот цикл в виде вектора H, записав его на место предыдущего хранящегося цикла, и перейти к шагу 4 или шагу 5.

Шаг 4. $B \leftarrow B - 1$, перейти к шагу 2.

Шаг 5. Останов. $m = B + 1$ является минимальным числом перенастроек аппаратуры, и последняя последовательность в H является требуемой последовательностью производства продуктов.

Шаг 1 приведенного выше алгоритма требует дальнейших пояснений. Очевидно, что чем более точной будет начальная верхняя граница B, тем меньшее число итераций основного алгоритма придется сделать. Процедура получения хорошей оценки для верхней границы дана в приложении. Существование гамильтонова цикла в графе G_B может быть установлено с использованием мультицепного алгоритма разд. 2.3.

ЧАСТЬ II

5. Задача коммивояжера

Задача коммивояжера тесно связана с несколькими другими задачами теории графов, обсуждаемыми в других частях этой книги. Здесь мы изучим две такие связи — с задачей о назначениях (см. гл. 12) и с задачей о кратчайшем остове (см. гл. 7). Обе эти задачи могут быть решены намного проще, чем задача коммивояжера, и можно исследовать эти связи для того, чтобы разработать эффективный метод решения последней задачи.

5.1. Нижняя граница из задачи о назначениях

Линейную задачу о назначениях для графа с произвольной матрицей весов $\mathbf{C} = [c_{ij}]$ можно сформулировать так (см. гл. 12).

Пусть $[\xi_{ij}]$ — $(n \times n)$-матрица, в которой $\xi_{ij} = 1$, если вершина x_i «назначена» к вершине x_j, и $\xi_{ij} = 0$, если x_i не назначена к x_j. Такую же схему можно использовать и в задаче коммивояжера, полагая $\xi_{ij} = 1$, если коммивояжер едет непосредственно из x_i в x_j, и $\xi_{ij} = 0$ в противном случае. В этой последней задаче можно положить $c_{ii} = \infty$ $(i = 1, \ldots, n)$, чтобы устранить бессмысленные решения с $\xi_{ij} = 1$.

Теперь задача о назначениях формулируется так.

Найти величины ξ_{ij}, минимизирующие

$$z = \sum_{j=1}^{n} \sum_{i=1}^{r} c_{ij}\xi_{ij}, \tag{10.4}$$

при условии, что

$$\sum_i \xi_{ij} = \sum_j \xi_{ij} = 1 \tag{10.5}$$

(для всех $i, j = 1, 2, \ldots, n$)

$$\xi_{ij} = 0 \text{ или } 1. \tag{10.6}$$

Условие (10—5) просто гарантирует, что решение будет циклическим, т. е. в каждую вершину входит и из нее выходит одна дуга.

Уравнения (10.4) — (10.6) вместе с дополнительным ограничением, состоящим в том, что решение должно давать *единственный* цикл (гамильтонов), а не просто некоторое число несвязных циклов также могут быть использованы для формулировки задачи коммивояжера. Заметим, что дополнительное условие $c_{ii} = \infty$

можно интерпретировать как ограничение, устраняющее возможность появления в решении задачи о назначениях циклов длины 1. Так как добавление любого ограничения в задаче о назначениях не может увеличить минимального значения z, определяемого из уравнений (10.4) — (10.6), то это значение z является нижней границей веса решения задачи коммивояжера для графа с матрицей весов $[c_{ij}]$.

5.2. Нижняя граница из задачи о кратчайшем остове

Для графа с симметричной матрицей весов (расстояний) **C**, т. е. для неориентированного графа, нижнюю границу для решения задачи коммивояжера можно получить по кратчайшему остову графа следующим образом. Пусть установлено, что ребро (x_1, x_2) входит в оптимальный цикл задачи коммивояжера. Если это ребро удалить из цикла, то получится цепь, состоящая из $n - 1$ ребер, проходящая через все вершины, начиная с вершины x_1, и оканчивающаяся в x_2. Так как вес кратчайшего остова, скажем L, является нижней границей для веса этой цепи, то длина кратчайшего остова плюс $c(x_1, x_2)$ будет нижней границей для веса оптимального решения задачи коммивояжера.

В общем случае заранее не известно, какие ребра содержатся в оптимальном цикле, но самое длинное ребро в цикле должно быть [15] не короче, чем $\max\limits_{x_i} [c(x_i, s)]$. Здесь s — вторая ближайшая вершина к вершине x_i. Таким образом, величина

$$L + \max_{x_i} [c(x_i, s)] \tag{10.7}$$

является нижней границей веса решения в задаче коммивояжера.

5.3. Двойственность

Определим G(TSP) как остовный подграф неориентированного графа G, образованный вершинами и теми ребрами графа G, которые использованы в оптимальном цикле коммивояжера. Аналогично определим графы G(AP) и G(SST), образованные теми же вершинами, но ребра которых берутся из оптимальных решений задачи о назначениях и задачи о кратчайшем остове соответственно.

Граф G(TSP) обладает следующими свойствами:

(1) граф является связным, т. е. каждую вершину можно соединить с любой другой вершиной цепью,

(2) степень каждой вершины равна 2, т. е. каждая вершина инцидентна двум ребрам.

На рис. 10.9 (а) приведен пример 5-вершинного графа G (TSP).

Граф G(AP) не обязательно обладает свойством (1), как это видно из рис. 10.9 (б), но по определению обладает свойством (2). Если, однако, окажется, что для решения задачи о назначениях выполняется и свойство (1), то это решение будет также решением задачи коммивояжера.

Для графа G(SST) свойство (1) выполняется по определению, но он может не иметь свойства (2). Если же окажется, что для кратчайшего остова свойство (2) выполняется — за исключением двух «конечных» вершин (скажем, x_1 и x_2), которые необходимо

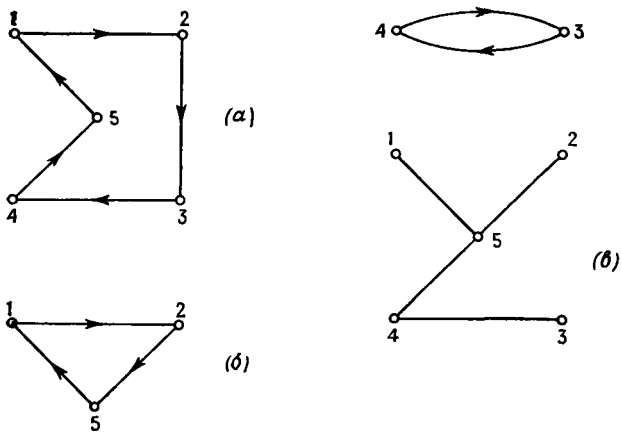


Рис. 10.9. Графы задач коммивояжера, о назначениях и о кратчайшем остове. (а) G (TSP). (б) G (AP). (в) G (SST).

имеют степень 1,— то кратчайший остов будет также кратчайшей цепью, проходящей через все n вершин. Если, кроме того, ребро (x_1, x_2) входит в оптимальный цикл задачи коммивояжера, то ребра кратчайшего остова вместе с ребром (x_1, x_2) дают решение задачи коммивояжера. (Если же (x_1, x_2) не обязательно принадлежит оптимальному циклу задачи коммивояжера, то необходима небольшая модификация; см. разд. 6.)

Таким образом, решения задач о назначениях и кратчайшем остове являются двойственными в том смысле, что они обладают свойствами, являющимися дополнительными по отношению к свойствам задачи коммивояжера. Теперь обнаруживаются два пути, могущие привести к решению последней задачи.

(I) Использовать решение задачи о назначениях (для которого выполняется свойство (2)) и добиться, чтобы это решение подчинялось свойству (1). (II) Использовать решение задачи о кратчайшем остове (для которого выполняется свойство (1)) и добиться выполнения свойства (2).

В разд. 6 этой главы мы исследуем путь, основанный на методе (II), а в в разд. 7 будет рассмотрен метод (I). Следует, однако, помнить, что хотя задача о назначениях определена для графов с произвольной структурой весов, кратчайший остов определяется только для неориентированных графов, т. е. для графов с симметричной ($c_{ij} = c_{ji}$) матрицей весов. Для несимметричных матриц весов (т. е. ориентированных графов) в гл. 7 было введено понятие древесности — аналог понятия остова. Таким образом, то, что было сказано о связи между задачей коммивояжера и кратчайшим остовом для неориентированных графов, имеет точный эквивалент, относящийся к связи между задачей коммивояжера и кратчайшей древесностью в случае ориентированных графов. В следующем разделе, однако, мы ограничимся симметричной задачей, так как распространение на более общий случай производится очевидным способом.

6. Задача коммивояжера и задача о кратчайшем остове

Как уже отмечалось в предыдущем разделе, симметричная задача коммивояжера тесно связана с задачей о кратчайшем остове графа G, и *открытая* задача коммивояжера (т. е. задача нахождения кратчайшей гамильтоновой *цепи* в графе) практически эквивалентна задаче нахождения кратчайшего остова графа G, с тем ограничением, что никакая вершина не должна иметь степень, большую чем 2. Ввиду того, что открытая задача коммивояжера несколько более сильно связана с задачей о кратчайшем остове, чем замкнутая, мы начнем с рассмотрения сначала этой задачи, отложив на конец раздела обсуждение тех небольших изменений, которые необходимо сделать в замкнутой (обычной) задаче коммивояжера.

Задача нахождения кратчайшей гамильтоновой цепи была впервые исследована Део и Хакими [13], давшими ее формулировку на языке линейного программирования. Для полного графа с n вершинами их формулировка содержит $n(n+1)$ переменных и $n(n+3)/2+1$ ограничений, которые формулируются явно. Наряду с этим имеется очень большое число ограничений, которые нельзя сформулировать явно, но которые рассматриваются неявно, причем за один раз (после каждого итерационного применения симплекс-метода) вводятся немногие из них. Хотя метод линейного программирования и дает всегда решение, он не будет здесь больше обсуждаться, так как обладает врожденной громоздкостью и неэффективностью.

6.1. Определения

Пусть $G = (X, A)$ — реберно-взвешенный неориентированный граф, а d_i^G — степень вершины x_i в графе G. Число вершин в G будет обозначаться через $n = |X|$.

Если задан остов $T = (X, A_T)$ графа G и степень вершины x_i в дереве T обозначена через d_i^T, можно определить близость этого остова к гамильтоновой цепи одним из следующих двух способов:

$$\varepsilon_T = \sum_{d_i^T > 2} (d_i^T - 2) \qquad (10.8(\text{а}))$$

или

$$\varepsilon_T = \sum_{i=1}^{n} |d_i^T - 2| - 2. \qquad (10.8(\text{б}))$$

Выражение (10.8)(а) учитывает близость только по тем вершинам, для которых $d_i^T > 2$, в то время как (10.8)(б) принимает во внимание и вершины степени 1. Оба эти выражения для гамильтоновой цепи дают $\varepsilon_T = 0$ и можно считать, что чем больше ε_T, тем сильнее дерево T отличается от гамильтоновой цепи.

Теперь мы рассмотрим две следующие задачи.

Задача (а). Кратчайшая гамильтонова цепь. Найти кратчайший остов $T^* \equiv (X, A^*)$ графа G, такой, что степени всех вершин не превышают 2.

Задача (б). Кратчайшая гамильтонова цепь с отмеченными концевыми вершинами. Заданы две вершины x_1 и x_2 ($x_1, x_2 \in X$), найти кратчайший остов $T_{1,2}^* \equiv (X, A_{1,2})$, такой, что степени всех вершин не превышают 2, а степени вершин x_1 и x_2 равны 1.

Теперь нужно проверить подразумеваемое в сформулированных задачах утверждение о том, что дерево T, все вершины которого имеют степень, не больше чем 2, является на самом деле гамильтоновой цепью. Действительно, так как T — дерево, то степень никакой его вершины не может быть равной нулю и, следовательно, $d_i^T = 1$ или 2 для всех i. Пусть q вершин имеют степень 1, а $n - q$ имеют степень 2. Число ребер в дереве равно

$$m_T = \frac{1}{2} \sum_{i=1}^{n} d_i^T, \qquad (10.9)$$

так как при суммировании каждое ребро считается дважды, по одному разу для каждой из его конечных вершин.

Из уравнения (10.9) получим

$$m_T = \frac{1}{2}[q + 2(n-q)] = n - \frac{q}{2}.$$

Так как число ребер в дереве равно $n-1$, то $q=2$ и поэтому ровно две вершины имеют степень 1, а $n-2$ вершин — степень 2, т. е. T является гамильтоновой цепью.

6.2. Алгоритм поиска, использующий дерево решений

6.2.1. Решение задачи (а). Мы рассмотрим сначала задачу (а) и объясним суть алгоритма, использующего дерево решений, на одном примере.

Пример. В качестве примера возьмем полный граф с 6 вершинами из работы Део и Хакими [13] со следующей матрицей весов:

$\mathbf{C}=[c_{ij}]=$

	1	2	3	4	5	6
1	0	4	10	18	5	10
2	4	0	12	8	2	6
3	10	12	0	4	18	16
4	18	8	4	0	14	6
5	5	2	18	14	0	16
6	10	6	16	6	16	0

С помощью метода гл. 7 легко найти кратчайший остов этого графа (без ограничений на d_i^T); он показан на рис. 10.10 (а). Из этого рисунка видно, что $d_2^T=3$, а так как нам нужен кратчайший остов, для которого $d_i^T\leqslant 2$, то можно сказать, что в окончательном ответе должно отсутствовать по крайней мере одно из ребер (2,1), (2,6) или (2,5). Таким образом, решение первоначальной задачи является решением по крайней мере одной из следующих трех частных задач, изображаемых узлами B, C и D дерева решений на рис. 10.11. На этом рисунке узел A представляет первоначальную задачу, а узлы B, C и D отвечают задачам, матрицы весов которых те же самые, что и матрица весов задачи A, но только ребрам (2,1), (2,6) или (2,5) соответственно приписан бесконечный вес.

На рис. 10.11 число, стоящее около узла, равно весу кратчайшего остова, соответствующего частной задаче, определяемой этим узлом. Следует заметить, что на любой стадии ветвления наименьший из весов у свободных узлов (т. е. узлов, в которых еще не произошло ветвление) является нижней границей для веса окончательного ответа.

Найдем теперь кратчайшие остовы $T(B)$, $T(C)$ и $T(D)$, соответствующие узлам B, C и D. Результаты изображены соответственно на рис. 10.10 (б), (в) и (г); веса этих остовов указаны вблизи узлов.

Остовы $T(B)$ и $T(D)$ являются гамильтоновыми цепями (все $d_i^T \leqslant 2$), в то время как $T(C)$ гамильтоновой цепью не является. Кратчайшим из этих двух остовов будет $T(B)$ с весом 23. Нижняя граница веса для окончательного ответа равна наименьшему из весов остовов $T(B)$, $T(C)$ и $T(D)$ и равна также 23. Таким

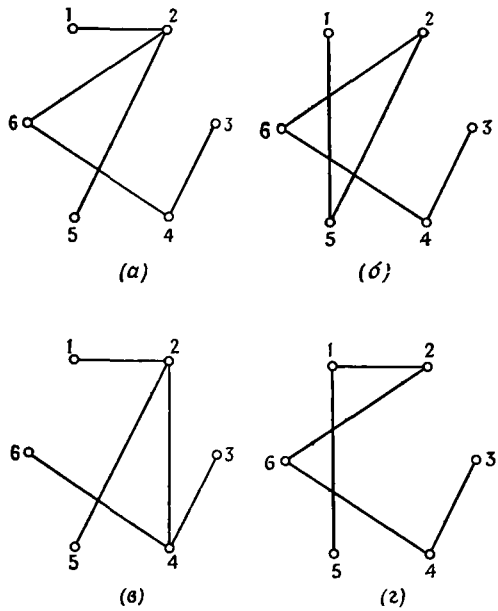


Рис. 10.10. Кратчайшие остовы в процессе поиска.
(а) $T^*(A)$, вес 22. (б) $T^*(B)$, вес 23. (в) $T^*(C)$, вес 24. (г) $T^*(D)$, вес 25.

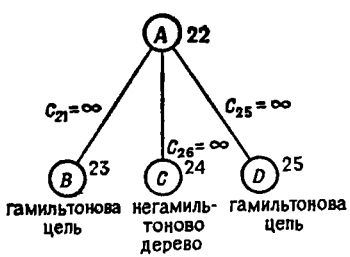


Рис. 10.11. Поиск с деревом решений.

образом, $T(B)$ дает оптимальный ответ, т. е. является кратчайшей гамильтоновой цепью. Следует заметить, что в узле C не нужно производить дальнейшее ветвление, так как любая гамильтонова цепь, полученная при таком ветвлении будет иметь вес, не меньший чем 24.

Итак, алгоритм, использующий дерево решений, позволил в этом частном примере получить оптимальный ответ, применяя алгоритм кратчайшего остова только 4 раза.

Другим побочным преимуществом данного метода является то, что он часто дает дополнительные гамильтоновы цепи, близкие к оптимальным, которые в практических задачах можно использовать в качестве альтернативного приемлемого решения.

6.2.2. Решение задачи (б). В этой задаче надо найти кратчайшую гамильтонову цепь с двумя заданными концевыми вершинами x_1 и x_2. Эту задачу можно свести к задаче (а), используя следующую теорему.

Теорема 2. *Пусть* $\mathbf{C} = [c_{ij}]$ *— матрица реберных весов первоначального графа* G *и* k *— достаточно большое положительное число, большее чем вес любой гамильтоновой цепи. Тогда решение задачи* (а) *с матрицей реберных весов* $\mathbf{C}'$, *где*

$$\left.\begin{aligned} c'_{j1} &= c_{j1} + k, \\ c'_{1j} &= c_{1j} + k, \\ c'_{2j} &= c_{2j} + k, \\ c'_{j2} &= c_{j2} + k, \end{aligned}\right\} \quad (\text{для всех } x_j \neq x_1, x_2)$$

$$\begin{aligned} c'_{ij} &= c_{ij} + 2k, && (\text{для всех } x_i, x_j = x_1, x_2) \\ c'_{ij} &= c_{ij}, && (\text{для всех } x_i, x_j \neq x_1, x_2) \end{aligned} \qquad (10.10)$$

является решением задачи (б) *с первоначальной матрицей весов.*

Доказательство. Каждая гамильтонова цепь относится к одной из следующих категорий:

(1) x_1 и x_2 не совпадают ни с одной из ее концевых вершин;

(2) одной из ее концевых вершин является или x_1, или x_2;

(3) одной концевой вершиной является x_1, а другой x_2. Вес гамильтоновой цепи с матрицей весов $\mathbf{C}'$ больше веса этой же цепи с матрицей $\mathbf{C}$ на величину:

$4k$, если цепь относится к категории (1);

$3k$, если цепь относится к категории (2);

$2k$, если цепь относится к категории (3).

Так как k превосходит вес любой гамильтоновой цепи, то вес (для матрицы $\mathbf{C}'$) самой длинной гамильтоновой цепи категории (3) меньше, чем вес самой короткой гамильтоновой цепи категории (2), а вес самой длинной гамильтоновой цепи категории (2) меньше, чем вес самой короткой цепи категории (1). Таким образом, решение задачи (а) с матрицей весов $\mathbf{C}'$ даст кратчайшую

гамильтонову цепь категории (3), т. е. она будет решением задачи (б) с матрицей весов **C**. Это доказывает теорему.

В связи с этим следует отметить, что относительные веса всех гамильтоновых цепей в пределах одной категории не изменяются при преобразовании (10.10).

6.3. Алгоритм штрафования вершин

В этом алгоритме используется та же идея, что и доказанная выше теорема 2, а именно такое преобразование матрицы весов **C**, при котором различные категории деревьев штрафуются поразному, но в то же время относительные веса всех деревьев в пределах одной категории не изменяются. Цель при этом состоит в том, чтобы разделить различные категории и сделать категорию гамильтоновых цепей более привлекательной (менее штрафуемой), а тогда применение алгоритма минимального дерева автоматически приведет к кратчайшей гамильтоновой цепи.

Теорема 3. *Если матрица* **C** *преобразована в матрицу* **C′** *так, что*

$$c'_{ij} = c_{ij} + p(i) + p(j) \quad \text{для} \quad i,\ j = 1,\ 2,\ \ldots,\ n, \qquad (10.11)$$

где $p(k)$ *— n-мерный вектор, образованный действительными положительными и отрицательными постоянными, то относительные веса всех гамильтоновых цепей для матрицы* **C′**, *имеющих указанные концевые вершины, не изменяются.*

Доказательство. Пусть F_C — вес произвольной гамильтоновой цепи с матрицей **C**, имеющей концевые вершины x_1 и x_2. Так как каждая вершина соединена точно с двумя другими вершинами (за исключением концевых вершин, каждая из которых соединена только с одной другой вершиной), то вес $F_{C'}$ той же самой гамильтоновой цепи с матрицей **C′** отличается от F_C на величину

$$F_{C'} - F_C = 2 \sum_{j \neq 1,\ 2} p(j) + p(1) + p(2). \qquad (10.12)$$

Это постоянная величина, не зависящая от выбранной гамильтоновой цепи. Отсюда следует теорема.

6.3.1. Решение задачи (б). Применение алгоритма нахождения кратчайшей гамильтоновой цепи с указанными концевыми вершинами (задача (б)), основанного на теореме 3, будет продемонстрировано на примере.

Пример. Рассмотрим полный граф G с 10 вершинами, матрица весов $\mathbf{C}_0$ которого дается табл. 10.1, и предположим, что мы хотим найти кратчайшую гамильтонову цепь с концевыми вершинами 8 и 9.

Таблица 10.1

Матрица весов примера

	1	2	3	4	5	6	7	8	9	10
1	0	28	31	28	22	36	50	67	40	74
2	28	0	31	40	41	64	74	80	63	101
3	31	31	0	14	53	53	53	50	42	83
4	28	40	14	0	50	41	39	41	28	69
5	22	41	53	50	0	40	61	86	53	78
6	36	64	53	41	40	0	24	58	22	39
7	50	74	53	39	61	24	0	37	11	30
8	67	80	50	41	86	58	37	0	36	60
9	40	63	42	28	53	22	11	36	0	41
10	74	101	83	69	78	39	30	60	41	0

В соответствии с теоремой 2 добавим к каждому элементу, стоящему либо в строках 8 и 9, либо в столбцах 8 и 9 таблицы 10.1, достаточно большое число k (скажем, 1000). Получившуюся после этого добавления таблицу назовем результирующей матрицей весов **C**.

Далее поступаем следующим образом.

Находим кратчайший остов графа **G**. Если он окажется гамильтоновой цепью, то задача решена. В рассматриваемом примере кратчайший остов, показанный на рис. 10.12 (а), не является гамильтоновой цепью. Степени вершин 1 и 7 равны 4, а не 2, как это требуется для гамильтоновой цепи. Следуя духу теоремы 3, мы можем «оштрафовать» эти вершины. Допустим, что мы произвольно выбрали малый шаг (скажем, 5 единиц), так что все штрафы кратны ему. (Можно выбрать величину шага равной 1, но это будет неоправдано мало и приведет к увеличению числа требуемых итераций.) Таким образом, если мы решили штрафовать только те вершины, степени которых больше чем 2, то

$$p(i) = 5(d_i^T - 2). \tag{10.13}$$

Заметим, что метод штрафования вершин выбран произвольно и существует много других альтернатив (см. разд. 6.3.4).

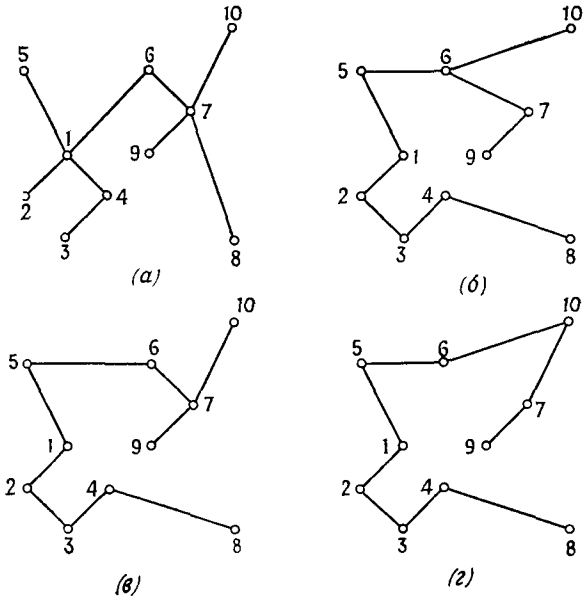


Рис. 10.12. Кратчайшие остовы при итерациях со штрафами.

В соответствии с уравнением (10.13) штрафы вершин 1 и 7 равны $p(1) = p(7) = 10$, причем для всех остальных вершин они равны 0. Следуя (10.11), вычислим новую матрицу весов и найдем для нее кратчайший остов. Результат показан на рис. 10.12 (б), из которого видно, что это дерево много ближе к гамильтоновой цепи, поскольку для него $\varepsilon = 1$ вместо $\varepsilon = 4$ в предыдущем случае.

Поступая как и ранее, оштрафуем вершину 6 и положим $p(6) = 5$. (Следует заметить, что этот новый штраф добавляется к предыдущей матрице весов, а не к первоначальной матрице $\mathbf{C}$.) Таким образом, штрафы равны $p(1) = p(7) = 10$, $p(6) = 5$ и $p(i) = 0$ для всех остальных вершин. Соответствующий кратчайший остов изображен на рис. 10.12 (в), и для него $\varepsilon = 1$, как и для предыдущего остова. Оштрафуем вершину 7, полагая $p(7) = 5$; новый кратчайший остов будет тем же, что и на рис. 10.12 (б). Продолжая далее, штрафуем вершину 6 еще раз, полагая $p(6) = 5$, и опять находим кратчайший остов. Результат показан на рис. 10.12 (г), из которого видно, что теперь это гамильтонова цепь и, следовательно, по теореме 3 — кратчайшая гамильтонова цепь. Вес этой цепи с первоначальной матрицей весов $\mathbf{C}_0$ равен 258.

Здесь полезно отметить, что если на некотором этапе получается кратчайший остов, уже встречавшийся ранее, то это не означает, что произошло зацикливание, и любой такой цикл автоматически устраняется при продолжении алгоритма. (Однако, как будет выяснено ниже, сходимость последовательности кратчайших остовов к кратчайшей гамильтоновой цепи не может быть гарантирована без конкретизации выбираемого способа штрафования вершин.) Мы получили кратчайшую гамильтонову цепь с концевыми вершинами 8 и 9 за 5 итераций, включая 5 вычислений кратчайших остовов, в то время как полный перебор содержал бы $8! = 40320$ вычислений матриц весов и сравнений цепей.

Альтернативой к вышеописанному методу штрафов является следующий. Вместо штрафования только тех вершин, для которых $d_i^T > 2$ (с использованием (10.13)), можно штрафовать только те вершины (за исключением двух выделенных конечных вершин), степени которых $d_i^T = 1 < 2$, причем величина штрафа $p(i)$ отрицательна и равна выбранной единице шага. С другой стороны, можно применить комбинацию этих двух политик наложения штрафов или воспользоваться совершенно другими политиками, описанными в разд. 6.3.4. Здесь следует отметить, что выбор политики наложения штрафов может существенно влиять на число необходимых для получения решения итераций.

Предположим в качестве примера, что на втором этапе описанных вычислений — когда кратчайший остов изображен на рис. 10.12 (б) — мы решили воспользоваться второй политикой наложения штрафов и вместо того, чтобы положить $p(6) = 5$, взяли $p(10) = -5$. Получившийся кратчайший остов был бы таким, какой показан на рис. 10.12 (г), т. е. он немедленно дал бы кратчайшую гамильтонову цепь.

Интересно отметить, что не существует единственного вектора штрафов $p(i)$, $i = 1, \ldots, n$, для которого матрица весов будет преобразовываться так, что кратчайший остов будет и гамильтоновой цепью и, следовательно, кратчайшей такой цепью. В приведенном примере окончательное значение вектора штрафов при первой политике такое: $p(1) = p(6) = 10$, $p(7) = 15$, все остальные $p(i) = 0$; в то же время при второй политике, принятой, начиная с шага 2, для окончательного вектора штрафов $p(1) = p(7) = 10$, $p(10) = -5$, все остальные $p(i) = 0$.

6.3.2. Решение задачи (а). Мы проиллюстрировали сначала применение алгоритма штрафования вершин к задаче (б), так как к этой задаче непосредственно применима теорема 3. Подобно тому как для решения задачи (б) с помощью алгоритма поиска, использующего дерево решений, она была сведена к задаче (а), так и сейчас, чтобы решить задачу (а) с помощью алгоритма

штрафования вершин, мы сведем ее к задаче (б). Это осуществляется посредством следующего приема.

К множеству X вершин графа G добавим еще две вершины, например x_1 и x_2, и образуем новое множество $X' = X \cup \{x_1, x_2\}$. К множеству ребер A графа G добавим два множества ребер:

$$S_1 = \bigcup_{j=1}^{n} \{(x_1, x_j)\} \quad \text{и} \quad S_2 = \bigcup_{j=1}^{n} \{(x_2, x_j)\},$$

и образуем новое множество $A' = A \cup S_1 \cup S_2$. Для всех $j = 1, \ldots, n$ зададим веса a — для ребер (x_1, x_j) и b — для ребер (x_2, x_j), где a и b — две произвольные постоянные.

Теорема 4. *Решение задачи* (б) *для графа* $G' = (X', A')$ *с концевыми вершинами* x_1 *и* x_2 *является решением задачи* (а) *для графа* G.

Доказательство. Любая гамильтонова цепь графа G' с концевыми вершинами x_1 и x_2 может рассматриваться как гамильтонова цепь, проходящая через вершины графа G с добавленными ребрами (x_1, x_i) и (x_2, x_j), $x_i, x_j \in X$. Таким образом, вес гамильтоновой цепи из G' равен, скажем, $F' = F + a + b$, где F — вес некоторой гамильтоновой цепи из G, и, следовательно, вес F' минимален только тогда, когда F минимален, т. е. когда F соответствует кратчайшей гамильтоновой цепи из G. Теорема доказана.

6.3.3. Сходимость метода штрафования вершин. Метод штрафования вершин, описанный выше, был развит Кристофидесом

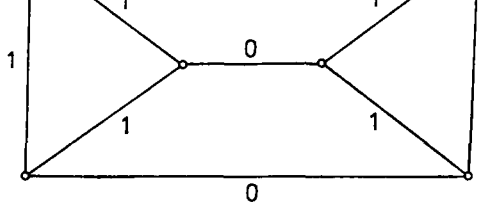


Рис. 10.13.

[7] и, в несколько иной форме, примерно в то же время Хелдом и Карпом [18]. Последние авторы показали также, что итерационный метод не обязательно сходится, и в качестве подходящего примера предложили граф, изображенный на рис. 10.13. Несмотря на этот большой недостаток, метод является тем не менее очень ценным по двум причинам.

Во-первых, в большинстве случаев он сходится (как будет показано ниже, все 33 случайно выбранные задачи были решены

этим методом по крайней мере при одной стратегии наложения штрафов).

Во-вторых, допустим, что итерации остановлены в некоторый момент, когда кратчайший остов с модифицированной матрицей есть T', штрафы вершин $p(i)$, $i = 1, 2, \ldots, n$, и степени вершин $d_i^{T'}$; рассматриваем задачу, когда нужно найти кратчайшую гамильтонову цепь с концевыми вершинами x_1 и x_2. Вес остова T' с модифицированной матрицей равен

$$\bar{F}_{T'} = F_{T'} + \sum_{i \neq 1,\ 2} p(i)\, d_i^{T'} + p(1) + p(2),$$

где $F_{T'}$ — вес остова T', вычисленный для первоначальной матрицы весов.

Если H — кратчайшая гамильтонова цепь графа, то вес цепи H с модифицированной матрицей равен

$$\bar{F}_H = F_H + 2 \sum_{i \neq 1,\ 2} p(i) + p(1) + p(2).$$

Так как по определению $\bar{F}_{T'} < \bar{F}_H$, то разность

$$f(p) \equiv \bar{F}_H - \bar{F}_{T'} = F_H - [F_{T'} + \sum_{i \neq 1,\ 2} p(i)\,(d_i^{T'} - 2)] \quad (10.14)$$

является мерой близости дерева T' к кратчайшей гамильтоновой цепи H, причем $f(\mathbf{p}) = 0$, если $T' = H$.

Величину $f(\mathbf{p})$ можно рассматривать как функцию вектора штрафов $\mathbf{p} = (p_i \mid i = 1, \ldots, n)$. Хелд и Карп предложили два метода нахождения штрафов $\mathbf{p}^*$, минимизирующих $f(\mathbf{p})$. Один основан на технике преобразования столбцов, а второй является методом наискорейшего спуска. (Другой, несколько более общий последовательный подход, основанный на эвристическом поиске, был недавно успешно применен Камерини и др. [4].) Если метод штрафования вершин сходится, то для минимального значения $f(\mathbf{p}^*) = 0$, так как $T' \to H$ и $d_i^{T'} = 2$ для всех $i \neq 1, 2$. С другой стороны, как было отмечено выше, метод сходится не всегда, и при отсутствии сходимости минимальное значение $f(\mathbf{p}^*)$ величины $f(\mathbf{p})$ равно некоторому положительному числу. Совершенно очевидно, что в этом случае величина

$$F_{T'} + \sum_{i \neq 1,\ 2} p^*(i)\,(d_i^{T'} - 2) \quad (10.15)$$

дает нижнюю границу веса кратчайшей гамильтоновой цепи. В этом последнем случае получаемая граница является очень точной и может быть использована с большим эффектом [19] в алгоритме поиска, применяющем дерево решений,— в духе того, как это делалось в разд. 6.2.1.

6.3.4. Стратегии наложения штрафов. Хотя, как уже отмечалось в разд. 6.3.3, существуют два метода [18] для определения последовательности штрафов, вынуждающих кратчайший остов становиться кратчайшей гамильтоновой цепью (если только вообще сходимость возможна), оба эти метода являются слишком изысканными и требуют больших затрат времени. В настоящем разделе мы экспериментально исследуем влияние различных стратегий штрафования вершин на скорость сходимости метода. Для всех 33 случайно выбранных полных графов, использованных для проверки стратегий, по крайней мере одна стратегия приводила к кратчайшей гамильтоновой цепи. Это убеждает нас в том, что, хотя и существуют задачи, в которых алгоритм штрафования вершин не сходится (на самом деле любой связный граф без гамильтонова цикла приводит нас к такой ситуации), но все же в большинстве задач, при составлении которых не старались специально найти контрпримеры, будет иметь место сходимость.

(I) *Фиксированные штрафы*

(a) *Только положительные штрафы*

Это тот случай, когда все вершины x_i дерева T со степенями $d_i^T > 2$ штрафуются на величину $\gamma \cdot (d_i^T - 2)$, а все остальные вершины не штрафуются. Было найдено, что для малых γ число необходимых для достижения решения итераций (т. е. определения некоторого гамильтонова цикла) приблизительно обратно пропорционально величине γ, но для достаточно больших γ вполне возможно зацикливание и решение получено не будет. Наилучшее значение γ зависит как от числа вершин, так и от распределения весов ребер.

(b) *Только отрицательные штрафы*

Это тот случай, когда вершины x_i (за исключением двух отмеченных концевых вершин) штрафуются на величину $-\gamma$, если $d_i^T = 1$, а остальные не штрафуются. Результаты примерно те же, что и в случае (a).

(c) *Положительные и отрицательные штрафы*

В этом случае вершины x_i штрафуются в соответствии с (a) и (b), в зависимости от того, будет ли $d_i^T > 2$ или $d_i^T = 1$ соответственно. Было найдено, что значение γ, равное половине используемого в случаях (a) и (b), обеспечивает сходимость примерно за то же самое число итераций, что и в случаях (a) и (b). Этот метод в общем лучше методов (a) и (b).

(II) Последовательно уменьшаемые штрафы

При использовании фиксированных штрафов может возникнуть ситуация, когда вершина x_i, с $d_i^T = 3$ (например), оштрафованная в некоторый момент на γ, превратится в вершину с $d_i^T = 1$ после следующей итерации, иными словами, вершина x_i «сверхоштрафована». Если используется смешанная стратегия (I (c)), то на следующей итерации вершина будет оштрафована на величину $-\gamma$, и это может привести к тому (не обязательно), что d_i^T снова станет равным 3. Из-за этих колебаний величины d_i^T может потребоваться неоправданно большое число итераций.

Было найдено, что в такой ситуации для большинства графов лучший метод состоит в уменьшении ранее наложенного штрафа, так что на следующей итерации для вышеприведенного примера следовало бы оштрафовать вершину x_i на $-\alpha\gamma$ вместо $-\gamma$, где $0 < \alpha < 1$.

Ситуация, аналогичная описанной, возникает и тогда, когда вершина со степенью $d_i^T = 1$ сверхштрафуется на отрицательную величину, в результате чего после очередной итерации она получает $d_i^T > 2$. В этом случае поступают как и в предыдущем.

В численных экспериментах, результаты которых приведены в табл. 10.2, было использовано значение $\alpha = 0{,}5$.

(III) Вычисляемые штрафы

(a) *Только положительные штрафы*

В этом случае все вершины x_i с $d_i^T > 2$ штрафуются на $p(i)$, где $p(i)$ вычисляется как минимальный положительный штраф, который, будучи применен к вершине x_i (изолированно), уменьшает d_i^T на одну единицу после одной итерации. Штрафы $p(i)$ применяются тогда одновременно — каждый к своей вершине.

Величина $p(i)$ для заданной вершины x_i вычисляется следующим образом. Удалим из дерева $T = (X, A_T)$ только одно из ребер (x_i, x_r), инцидентных вершине x_i. Это приведет к распадению T на два поддерева T_1 и T_2. Найдем ребро наименьшего веса, соединяющее эти два поддерева, т. е. найдем ребро (x_j^r, x_k^r) с весом

$$c(x_j^r, x_k^r) = \min_{\substack{x_j \in T_1 \\ \neq x_i}} \min_{\substack{x_k \in T_2 \\ \neq x_l}} [c(x_j, x_k)], \qquad (10.16)$$

где x_j^r и x_k^r — оптимальные значения x_j и x_k соответственно, а T_1 и T_2 обозначают как деревья, так и соответствующие множества их вершин. Вес $c(x_j^r, x_k^r)$ является поэтому наименьшим весом «связи» поддеревьев T_1 и T_2 в одно дерево, когда удалено первоначальное ребро (x_i, x_r). Таким образом, штраф $c(x_j^r, x_k^r)$ —

$c(x_i, x_r)$, налагаемый на вершину x_i, является наименьшим штрафом, который удалил бы ребро (x_i, x_r) из кратчайшего остова при следующей итерации. Поэтому если штраф выбран как

$$p(i) = \min_{(x_i, x_r) \in A_T} [c(x_j^r, x_k^r) - c(x_i, x_r)], \qquad (10.17)$$

то штраф $p(i)$, наложенный только на вершину x_i, удалил бы (в предположении, что $p(i)$ является единственным) только одно ребро (x_i, x_r^*) из множества ребер, инцидентных x_i, т. е. уменьшил бы d_i^T на единицу. Ребро (x_i, x_r^*) является как раз тем ребром, которое минимизирует выражение (10.17).

Здесь следует отметить, что когда все штрафы $p(i)$, вычисленные по (10.17), накладываются одновременно каждый на свою вершину, степени некоторых вершин могут уменьшиться более чем на 1, в то время как степени других вершин могут вообще не уменьшиться (или могут даже увеличиться). Это объясняется совместным влиянием штрафов вершин на веса ребер при следующей итерации. Таким образом, хотя использование штрафов, найденных из (10.17), не гарантирует получение ответа после заданного числа итераций, но, как было обнаружено (см. табл. 10.2), этот метод штрафования лучше, чем методы (I) и (II).

(b) *Только отрицательные штрафы*

Это тот случай, когда все вершины x_i с $d_i^T = 1$ штрафуются на отрицательную величину $p(i)$, где $p(i)$ вычисляется как максимальный (наименьший отрицательный) штраф, который, будучи применен только к вершине x_i, сделает степень d_i^T равной 2 после одной итерации, т. е. добавит второе ребро к вершине x_i. Однако штрафы $p(i)$ применяются одновременно — каждый к своей вершине.

Значение $p(i)$ для данной вершины x_i вычисляется следующим образом. Добавим ребро (x_i, x_r) к дереву T. Это добавление приведет к замыканию цикла, составленного из ребра (x_i, x_r) вместе с ребрами дерева T, входящими в цепь от x_r до x_i. Пусть множество ребер дерева T в цепи от x_r до x_i (исключая последнее ребро, инцидентное x_i) будет S_{ri}. Если тогда добавить ребро (x_i, x_r) к ребрам T и удалить одно какое-либо ребро из S_{ri}, то получится другое дерево, в котором степень вершины x_i равна двум. Если, таким образом, накладывается единственный штраф $p(i)$ на вершину x_i, где

$$p(i) = \min_{x_r \in X} [c(x_i, x_r) - \min_{(x_j, x_k) \in S_{ri}} \{c(x_j, x_k)\}], \qquad (10.18)$$

(т. е. $p(i)$ равен наименьшему дополнительному весу добавления ребра от x_i к какой-нибудь другой вершине x_r и удаления ребра

Таблица 10.2

Вычислительные результаты † в методе штрафования вершин

Число вершин n	*Стратегия штрафования*													
	I(a)		I(b)		I(c)		II		III(a)		III(b)		III(c)	
	α	β	α	β	α	β	α	β	α	β	α	β	α	β
10	22	0,15	16	0,13	18	0,11	16	0,18	71	0,18	27	0,16	9	0,12
16	39	0,27	33	0,24	35	0,29	31	0,20	52	0,42	18	0,39	22	0,50
20	83	0,88	56	0,71	74	0,80	39	0,38	55	0,59	24	0,48	27	0,67
26	126	1,83	158	1,41	98	1,66	74	1,02	40	0,47	60	0,71	16	0,46
30	—	—	—	—	—	—	—	—	59	0,91	63	1,11	61	0,95
36									98	1,41	87	1,63	72	1,37
40									172	2,23	159	1,99	57	1,00
46									—	—	—	—	81	3,09
50													92	4,43
56													146	9,70
60													111	13,6

α: *Число итераций*

β *Время вычисления (CDC 6600)*

† *Каждый элемент является средним значением для 3 графов одного и того же размера*

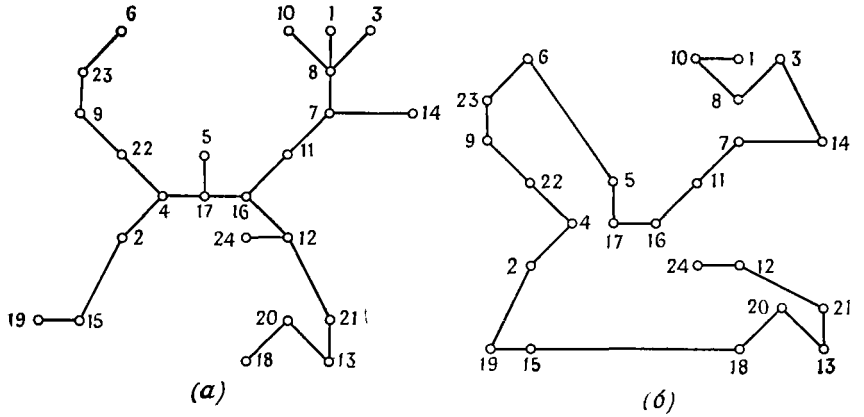


Рис. 10.14. Задача с 24 вершинами. (а) Начальный кратчайший остов. (б) Кратчайший гамильтонов цикл, полученный после 40 итераций с использованием штрафов III(c).

наименьшего веса в цепи от x_r к x_i), то степень вершины x_i станет равной 2 после единственной итерации.

Если все штрафы $p(i)$ накладываются одновременно, то, как и в случае (III (a)), взаимодействие этих штрафов делает невозможным предсказание изменения степеней вершин после одной итерации.

(c) *Положительные и отрицательные штрафы*

Эта стратегия заключается в наложении штрафов на вершины в соответствии с (III (a)) или (III (b)) в зависимости от того, будет ли $d_i^T > 2$ или $d_i^T = 1$.

В таблице 10.2 приводятся результаты и делается сравнение семи стратегий штрафования, описанных выше. Графы в этой таблице были получены случайным выбором n точек в квадрате при равномерном распределении, а в качестве весов ребер c_{ij} бралось евклидово расстояние между i и j. Таблица дает числа итераций и времена вычислений (ЭВМ CDC 6600, время в секундах), необходимые для получения ответа. Из этой таблицы видно, что действие всех стратегий штрафования зависит от задачи, хотя можно заметить, что наилучшей является стратегия (III (c)). Используя эту стратегию, можно найти кратчайшую гамильтонову цепь в графе примерно с 60 вершинами менее чем за 15 с.

На рис. 10.14 для некоторого графа с 24 вершинами показаны начальный кратчайший остов и окончательная кратчайшая гамильтонова цепь. Первая и последняя пронумерованные вершины являются отмеченными концевыми вершинами.

6.4. Задачи, родственные задаче коммивояжера

До сих пор в разд. 6 мы имели дело с «открытой» задачей коммивояжера, т. е. с кратчайшей гамильтоновой цепью (а не с циклом). Вначале это было оправдано тем, что при этом мы имели дело с более прямой связью «открытой» задачи с кратчайшим остовом. С другой стороны, нужно сделать только очень небольшие изменения, чтобы подойти и к «замкнутой» задаче коммивояжера. Хелд и Карп [18], например, ввели понятие кратчайшего 1-дерева графа G, причем оно определяется как кратчайший остов порожденного подграфа графа G с удаленной вершиной 1 плюс два кратчайших ребра, исходящих из вершины 1 к двум другим вершинам дерева. Очевидно, что между понятием кратчайшего 1-дерева и «замкнутой» задачей коммивояжера существует та же самая связь, что и между понятием кратчайшего остова и «открытой» задачей. Таким образом, штрафование вершин изменяет относительные веса 1-деревьев, но оставляет инвариантным относительное упорядочение гамильтоновых циклов. Так же совершенно очевидно, что, как в «открытой» задаче, кратчайший остов со всеми вершинами степени 2 (за исключением двух концевых вершин) становится кратчайшей гамильтоновой цепью между этими концевыми вершинами, так и в «замкнутой» задаче кратчайшее 1-дерево, все вершины которого имеют степень 2, является кратчайшим гамильтоновым циклом графа. Алгоритм штрафования вершин, обсуждавшийся ранее в данном разделе, может быть использован поэтому фактически без изменений и для решения «замкнутой» задачи коммивояжера.

7. Задача коммивояжера и задача о назначениях

В разд. 5 было отмечено, что задача о назначениях, определяемая соотношениями (10.4), (10.5) и (10.6), может иметь решения, образованные некоторым числом непересекающихся циклов, и что соответствующий метод решения задачи коммивояжера требует наложения ограничений до тех пор, пока решение не будет состоять из единственного (гамильтонова) цикла. В настоящем разделе мы исследуем процедуры наложения этих ограничений в рамках алгоритма поиска, использующего дерево решений. Применяемая схема будет в основном такой же, как и в разд. 6.2 для алгоритма поиска, основанного на рассмотрении кратчайшего остова.

7.1. Алгоритм поиска, использующий дерево решений

Пусть решение задачи о назначениях с матрицей весов $[c_{ij}]$ (где $c_{ii} = \infty$, $\forall i$) образовано некоторым числом непересекающихся циклов. Так, например, если решение задачи с 8 вершинами имеет вид $\xi_{12} = \xi_{26} = \xi_{35} = \xi_{47} = \xi_{54} = \xi_{61} = \xi_{78} = \xi_{83} = 1$ и $\xi_{ij} = 0$ для всех остальных пар (i, j), то решение, отвечающее двум циклам, изображено на рис. 10.15 (а). Теперь необходимо

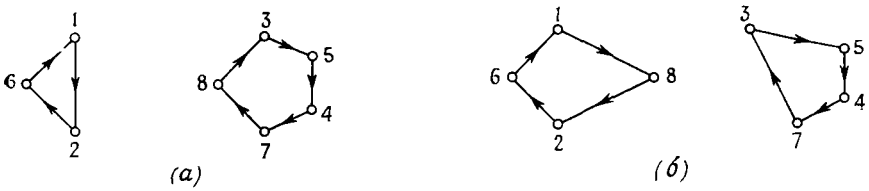


Рис. 10.15. Решение задач о назначениях. (а) Решение задачи P_0. (б) Решение задачи P_1.

сделать следующее — исключить данное решение вместе со многими другими такими же решениями, не потеряв решение задачи коммивояжера с той же матрицей весов. Так как решение задачи коммивояжера является гамильтоновым циклом, то мы будем пытаться исключить любое решение, состоящее более чем из одного цикла.

(А) Правило простого ветвления. Пусть в общем случае решение задачи о назначениях содержит (не гамильтонов) цикл $(x_1, x_2, \ldots, x_k, x_1)$ длины k. Удаление этого цикла (и всех решений, его содержащих) из дальнейшего рассмотрения может быть произведено путем наложения требования, что по крайней мере одна из дуг (x_1, x_2), (x_2, x_3), $\ldots$, (x_k, x_1) не должна входить в решение. Это можно сделать совсем просто, если первоначальную задачу с матрицей весов $[c_{ij}]$ разбить на k подзадач $P_1, P_2, \ldots, P_k$. В задаче P_1 $c(x_1, x_2)$ полагается равным ∞, а все остальные c_{ij} остаются без изменения (т. е. такими же, как и в задаче P_0), в задаче P_2 $c(x_2, x_3) = \infty$ и вообще в задаче P_k $c(x_k, x_1) = \infty$. Очевидно, что решение задачи P_0, не содержащее цикла $(x_1, x_2, \ldots, x_k, x_1)$, является решением по крайней мере одной из задач $P_1, \ldots, P_k$ и, следовательно, оптимальное решение задачи коммивояжера является решением одной или нескольких из этих подзадач.

Так, например, исключая на рис. 10.15 (а) цикл длины 3, получим дерево решений, изображенное на рис. 10.16, в котором задачи P_1, P_2 и P_3 представлены узлами дерева, растущего из начальной задачи P_0. Пусть задачи P_1, P_2 и P_3 решены как задачи о назначениях, и пусть соответствующие веса будут C_1, C_2 и C_3. Так как C_1 является нижней границей в задаче коммивояжера

для P_1, а также для P_2 и P_3, то число $L = \min(C_1, C_2, C_3)$ является нижней границей для веса решения первоначальной задачи коммивояжера.

Пусть, скажем, $L = C_1$ (т. е. $C_1 \leqslant C_2$ и $C_1 \leqslant C_3$). Если решение задачи P_1 является гамильтоновым циклом, то оно будет решением первоначальной задачи коммивояжера. В противном случае пусть оно будет, скажем, таким, как на рис. 10.15 (б). Исключая цикл (1, 8, 2, 6, 1), мы снова составляем и решаем подзадачи P_4, P_5, P_6 и P_7, показанные на рис. 10.16. Из этого

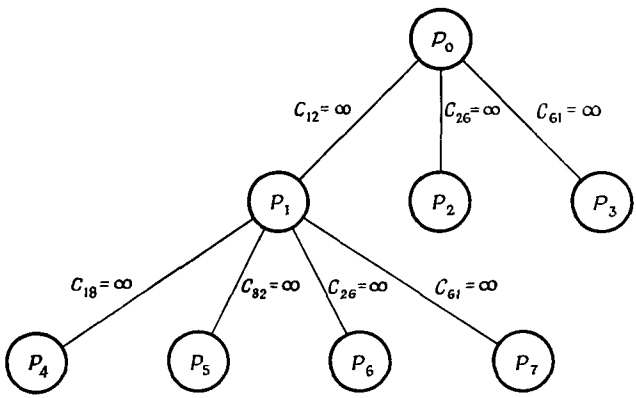


Рис. 10.16. Дерево решений с простым правилом ветвления А.

рисунка видно, что P_4 отвечает задаче, для которой в матрице весов элемент c_{18} положен равным ∞ (в дополнение к тому, что элемент c_{12} положен равным ∞ уже ранее). То же самое относится и к P_5, P_6 и P_7. Нижняя граница теперь переопределяется как $L = \min(C_2, C_3, C_4, C_5, C_6, C_7)$. Допустим, что задача P_3 соответствует весу L (т. е. $L = C_3$). Если решение задачи P_3 является гамильтоновым циклом, то оно будет решением первоначальной задачи коммивояжера. Дальнейшее ветвление в узле P_3, с другой стороны, должно осуществляться точно так же, как это делалось в P_1, и этот процесс продолжается до тех пор, пока задача с текущим значением веса L не будет иметь в качестве решения гамильтонов цикл. Когда это произойдет, то полученный цикл будет окончательным решением, так как его вес (по определению числа L) не превосходит нижних границ весов любых других гамильтоновых циклов, которые могут появиться при дальнейшем ветвлении в оставшихся узлах дерева.

Из приведенного описания алгоритма становится очевидным, что примененный поиск с помощью дерева решений относится к типу с «приоритетом по ширине», как это объяснено в приложении 1.

(Б) Правило исключающего ветвления. Как объяснено в приложении 1, для хорошего ветвления при разбиении задачи P_0 на подзадачи P_1, P_2 и P_3 требуется только, чтобы каждое возможное решение задачи P_0 (за исключением удаляемых решений) было решением по крайней мере одной из подзадач. Однако, как отмечено в приложении, очень желательным свойством метода ветвления было бы разбиение возникающих подзадач на взаимно исключающие, коль скоро речь идет о возможных решениях задачи P_0. Иными словами, каждое допустимое решение задачи P_k должно быть решением одной, и только одной из этих подзадач.

Ранее описанное правило ветвления основывалось на том факте, что цикл, такой, как $(x_1, x_2, \ldots, x_k, x_1)$, может быть удален посредством исключения одной из его дуг. Это, однако, не приводит в процессе ветвления к взаимно исключающим подзадачам. Так, в вышеприведенном примере решение задачи P_0, соответствующее циклам (1, 3, 6, 1) и (2, 5, 4, 7, 8, 2), является допустимым решением как подзадачи P_1 (с $c_{12} = \infty$), так и P_2 (с $c_{26} = \infty$).

Другим правилом ветвления, удаляющим цикл $(x_1, x_2, \ldots, x_k, x_1)$ и приводящим к взаимно исключающим подзадачам, является следующее:

для задачи P_1 положить $c(x_1, x_2) = \infty$;

для задачи P_2 положить $c(x_1, x_2) = -M$ и $c(x_2, x_3) = \infty$;

для задачи P_3 положить $c(x_1, x_2) = c(x_2, x_3) = -M$ и $c(x_3, x_4) = \infty$.

. . . . . .

для задачи P_k положить $c(x_1, x_2) = c(x_2, x_3) = \ldots = c(x_{k-1}, x_k) = -M$ и $c(x_k, x_1) = \infty$,

где $-M$ — достаточно большое отрицательное число, гарантирующее, что дуга, вес которой равен $-M$, входит в оптимальное решение.

Это правило ветвления наверняка приводит к взаимно исключающим подзадачам, так как для любых двух подзадач имеется по крайней мере одна дуга, исключенная из решения одной, но заведомо входящая в решение другой подзадачи. Легко также видеть, что ни одно возможное решение задачи P_0 не будет потеряно, т. е. что любое решение первоначальной задачи P_0 будет являться решением одной из подзадач. Это очевидно в силу того, что любое решение задачи P_0 имеет некоторую последовательность дуг, выходящую из x_1, таких, как (x_1, x_α), (x_α, x_β) и т. д., и первые r дуг должны совпадать с дугами цепи $(x_1, x_2, x_3, \ldots, x_k)$ при некотором значении $r = 0, 1, \ldots, k$. Значение $r = 0$ отвечает тому случаю, когда совпадения вообще нет, $r = 1$ — случаю, когда $x_\alpha = x_2$, но $x_\beta \neq x_3$ и т. д.

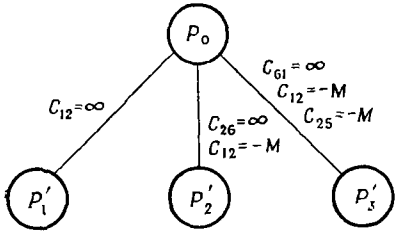


Рис. 10.17. Дерево решений с исключающим правилом ветвления Б.

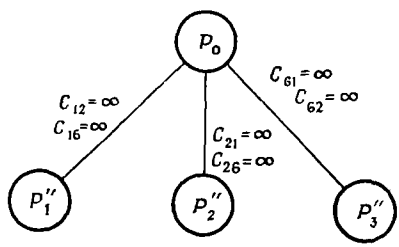


Рис. 10.18. Дерево решений с улучшенным правилом ветвления В.

В приведенном примере первоначальная задача P_0 разбивалась бы на три подзадачи, показанные на рис. 10.17 (ср. с разбиением на первом уровне из рис. 10.16, полученным по предыдущему правилу ветвления).

(В) Лучшее правило ветвления. Оба предыдущих правила ветвления исключали (при каждом отдельном ветвлении) все решения, содержащие заданный цикл, такой, как $(x_1, x_2, \ldots, x_k, x_1)$. Очевидно, однако, что в решении задачи коммивояжера не только не должен не существовать такой цикл, но должна быть по крайней мере одна дуга, ведущая из множества вершин $S = \{x_1, \ldots, x_k\}$ в множество вершин $\bar{S} = X - S$. Действительно, существование какой-либо дуги, ведущей из S в $\bar{S}$, не только гарантирует исключение решений, содержащих цикл, принадлежащий множеству S, но также позволяет исключить решения, множества вершин которых лежат в S и которые состоят из нескольких циклов (а не только из одного). Таким образом, можно ожидать, что правило ветвления, основанное на требовании, чтобы существовала некоторая дуга из S в $\bar{S}$, будет равномерно лучше, чем два предыдущие правила ветвления 2.

Так как дуга из S в $\bar{S}$ должна начинаться в некоторой вершине из S, то задачу можно расщепить на k подзадач. В подзадаче P_i мы будем требовать, чтобы начальной вершиной дуги являлась $x_i \in S$, а конечной вершиной была некоторая вершина в $\bar{S}$. Это можно сделать, положив $c(x_i, x_j) = \infty$ для всех $x_i \in S$ и оставив без изменения все другие веса. В решении получившейся задачи о назначениях мы наверняка бы имели дугу из x_i, ведущую в $\bar{S}$, так как для всех других альтернатив их веса должны быть положены равными ∞.

В примере, данном выше, начальная задача P_0 была бы в соответствии с настоящим правилом ветвления подразделена на три подзадачи, определенные, как показано на рис. 10.18. Дальнейшие ветвления происходят аналогично.

7.2. Пример

Найти решение задачи коммивояжера с матрицей весов

$C_A =$

	1	2	3	4	5	6	7	8
1	∞	76	43	38	51	42	19	80
2	42	∞	49	26	78	52	39	87
3	48	28	∞	36	53	44	68	61
4	72	31	29	∞	42	49	50	38
5	30	52	38	47	∞	64	75	82
6	66	51	83	51	22	∞	37	71
7	77	62	93	54	69	38	∞	26
8	42	58	66	76	41	52	83	∞

Теперь мы используем только что описанный алгоритм исключения циклов, беря на каждом шаге циклы наименьшей длины и используя правило ветвления B.

Решение задачи о назначениях с матрицей C_A состоит из двух циклов (2, 4, 3) и (1, 7, 8, 6, 5) с весом 232. Это соответствует

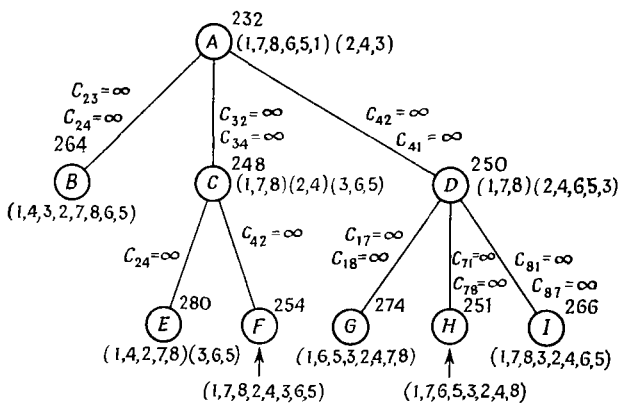


Рис. 10.19. Поиск, использующий дерево решений из примера 7.2.

узлу A дерева решений, показанного на рис. 10.19. Исключение цикла (2, 4, 3) по правилу ветвления B приводит к трем подзадачам, изображаемых на рис. 10.19 узлами B, C и D. Матрицы весов

этих подзадач, решения задач о назначениях для них и веса этих решений равны соответственно

$\mathbf{C}_B =$

	1	2	3	4	5	6	7	8
1	∞	76	43	38	51	42	19	80
2	42	∞	∞	∞	78	52	39	87
3	48	28	∞	36	53	44	68	61
4	72	31	29	∞	42	49	50	38
5	30	52	38	47	∞	64	75	82
6	66	51	83	51	22	∞	37	71
7	77	62	93	54	69	38	∞	26
8	42	58	66	76	41	52	83	∞

(1, 4, 3, 2, 7, 8, 6, 5)
264

$\mathbf{C}_C =$

	1	2	3	4	5	6	7	8
1	∞	76	43	38	51	42	19	80
2	42	∞	49	26	78	52	39	87
3	48	∞	∞	∞	53	44	68	61
4	72	31	29	∞	42	49	50	38
5	30	52	38	47	∞	64	75	82
6	66	51	83	51	22	∞	37	71
7	77	62	93	54	69	38	∞	26
8	42	58	66	76	41	52	83	∞

(1, 7, 8), (2, 4), (3, 6, 5)
248

Решение подзадачи B является гамильтоновым циклом с весом 264. Однако нижние границы в обоих узлах C и D меньше этого значения, так что существует возможность получить лучшее реше-

$C_D =$

	1	2	3	4	5	6	7	8
1	∞	76	43	38	51	42	19	80
2	42	∞	49	26	78	52	39	87
3	48	28	∞	36	53	44	68	61
4	72	∞	∞	∞	42	49	50	38
5	30	52	38	47	∞	64	75	82
6	66	51	83	51	22	∞	37	71
7	77	62	93	54	69	38	∞	26
8	42	58	66	76	41	52	83	∞

(1, 7, 8), (2, 4, 6, 5, 3)
250

ние, продолжая процесс ветвления. Наименьшая из нижних границ есть $L = \min(248, 250) = 248$, и она соответствует узлу C, поэтому процесс ветвления следует продолжать с этого узла, исключая цикл (2, 4). Теперь возникают две другие подзадачи, показанные узлами E и F на рис. 10.19. Их матрицы весов, решения задач о назначениях и веса таковы:

$C_E =$

	1	2	3	4	5	6	7	8
1	∞	76	43	38	51	42	19	80
2	42	∞	49	∞	78	52	39	87
3	48	∞	∞	∞	53	44	68	61
4	72	31	29	∞	42	49	50	38
5	30	52	38	47	∞	64	75	82
6	66	51	83	51	22	∞	37	71
7	77	62	93	54	69	38	∞	26
8	42	58	66	76	41	52	83	∞

(3, 6, 5), (1, 4, 2, 7, 8)
280

$\mathbf{C}_F =$

	1	2	3	4	5	6	7	8
1	∞	76	43	38	51	42	19	80
2	42	∞	49	26	78	52	39	87
3	48	∞	∞	∞	53	44	68	61
4	72	∞	29	∞	42	49	50	38
5	30	52	38	47	∞	64	75	82
6	66	51	83	51	22	∞	37	71
7	77	62	93	54	69	38	∞	26
8	42	58	66	76	41	52	83	∞

(1, 7, 8, 2, 4, 3, 6, 5)
254

Следует отметить, что на этом этапе решение подзадачи *F* *является* гамильтоновым циклом с весом 254. Этот вес меньше, чем вес предыдущего лучшего решения (с весом 264), и, следовательно, лучшим текущим решением будет решение подзадачи *F*. Два узла (*E* и *D*) все еще дают решения с негамильтоновыми циклами и являются поэтому кандидатами на продолжение процесса ветвления. Однако для узла *E* нижняя граница равна $280 > 254$ (текущее значение наилучшего решения), и, значит, ветвление в этом узле не может привести к лучшему результату. Поэтому

$\mathbf{C}_G =$

	1	2	3	4	5	6	7	8
1	∞	76	43	38	51	42	∞	∞
2	42	∞	49	26	78	52	39	87
3	48	28	∞	36	53	44	68	61
4	72	∞	∞	∞	42	49	50	38
5	30	52	38	47	∞	64	75	82
6	66	51	83	51	22	∞	37	71
7	77	62	93	54	69	38	∞	26
8	42	58	66	76	41	52	83	∞

(1, 6, 5, 3, 2, 4, 7, 8)
274

узел D с нижней границей $250 < 254$ является единственным узлом, ветвление в котором может привести к улучшению. Исключение цикла (1, 7, 8) из решения для узла D приводит к трем подзадачам, обозначенным на рис. 10.19 как G, H и I. Соответствующие этим подзадачам матрицы весов и решения задач о назначениях таковы:

$\mathbf{C}_H =$

	1	2	3	4	5	6	7	8
1	∞	76	43	38	51	42	19	80
2	42	∞	49	26	78	52	39	87
3	48	28	∞	36	53	44	68	61
4	72	∞	∞	∞	42	49	50	38
5	30	52	38	47	∞	64	75	82
6	66	51	83	51	22	∞	37	71
7	∞	62	93	54	69	38	∞	∞
8	42	58	66	76	41	52	83	∞

(1,7,6,5,3,2,4,8)
251

$\mathbf{C}_I =$

	1	2	3	4	5	6	7	8
1	∞	76	43	38	51	42	19	80
2	42	∞	49	26	78	52	39	87
3	48	28	∞	36	53	44	68	61
4	72	∞	∞	∞	42	49	50	38
5	30	52	38	47	∞	64	75	82
6	66	51	83	51	22	∞	37	71
7	77	62	93	54	69	38	∞	26
8	∞	58	66	76	41	52	∞	∞

(1,7,8,3,2,4,6,5)
266

На этом этапе мы замечаем, что решение подзадачи H является гамильтоновым циклом с весом 251, что меньше чем предыдущее

лучшее значение 254. Поэтому решение подзадачи H, т. е. (1, 7, 6, 5, 3, 2, 4, 8), заменяет предыдущее лучшее решение. Кроме того, значение 251 меньше, чем нижняя граница в любом конечном узле дерева, и, следовательно, найдено оптимальное решение всей задачи. (Решения во всех концевых узлах — за исключением узла E, ветвление в котором было прекращено ранее,— являются на самом деле гамильтоновыми циклами, и здесь безотносительно к границам весов этих решений не нужно бы было производить дальнейшее ветвление.)

7.3. Вычислительные комментарии и характеристики

Только что описанный алгоритм поиска, использующий дерево решений и основанный на исключении циклов, зависит от нахождения решений задач о назначениях с очень незначительно измененными матрицами весов. Здесь следует отметить, что каждая из этих задач может быть решена очень просто, если хранить решение предыдущей задачи, из которой она возникла. В частности, модификации, производимые в случае всех трех ранее упомянутых правил ветвления, производятся с помощью приравнивания ∞ некоторого элемента $c(x_i, x_j)$ для дуги (x_i, x_j), входящей в решение текущей задачи о назначениях.

В венгерском алгоритме (см. гл. 12) решения задачи о назначениях элемент $c(x_i, x_j)$ в окончательной относительной матрице весов должен был бы иметь значение 0, а соответствующий маркер показывает, что в решении было назначение. Изменение значения элемента $c(x_i, x_j)$ привело бы, очевидно, к перераспределению этого назначения, но оставило бы без изменения другие $n-1$ назначений. Таким образом, начиная с решения (назначений) задачи — до того как были произведены модификации матрицы $[c_{ij}]$ — и удаляя упомянутое назначение, можно получить решение новой модифицированной задачи, повторно применяя венгерский алгоритм на последнем шаге, так как единственный «прорыв», т. е. единственное увеличение числа ненулевых назначений с $n-1$ до n, дал бы в действительности оптимальное решение новой задачи о назначениях (детали см. в гл. 12).

Рабочие качества вышеописанного алгоритма поиска с исключением циклов по правилу ветвления B исследовали Беллмор и Мэлоун [2]. На любой стадии ветвление в узле продолжалось так, чтобы исключить циклы наименьшей длины в решении задачи о назначениях, соответствующей этому узлу. Задача коммивояжера с произвольной асимметричной матрицей весов может быть решена на ИБМ 7094 II за T секунд, где

$$T \approx 0{,}55 \cdot 10^{-4} \cdot n^{3{,}46}$$

и n — число вершин в задаче. (Было показано, что два других правила ветвления обладают худшими характеристиками, особенно

в задачах, графы которых содержат такие «скопления» вершин, что дуги в одном и том же скоплении имеют малые веса, а дуги между вершинами из разных скоплений имеют большие веса.)

Тем не менее вышеописанный алгоритм работает не слишком хорошо в симметричных задачах, так как в них решения задач о назначениях обычно состоят из большого числа циклов длины 2 и для их исключения требуется очень много ветвлений. Еще хуже обстоит дело тогда, когда задача коммивояжера решается на графе, который не содержит гамильтоновых циклов конечного веса.

7.4. Лучшие границы для дерева поиска

Мы описали в разд. 7.1 алгоритм поиска, в котором в качестве нижней границы в узле берется значение из решения соответствующей задачи о назначениях. На самом-то деле совершенно очевидно, что метод исключения циклов остается неизменным независимо от того, какая нижняя граница используется для ограничения поиска. Но так как процесс ветвления в узле прекращается только тогда, когда или решение подзадачи, отвечающей этому узлу, является гамильтоновым циклом, или когда нижняя граница в узле превосходит значение полученного до этого наилучшего решения, то очевидно, что качество оценки границы оказывает заметное влияние на число ветвлений в дереве решений и, следовательно, на вычислительную эффективность метода. Целью настоящего раздела является описание метода получения нижней границы, вычисляемой по решению задачи о назначениях с небольшими дополнительными усилиями и являющейся более точной, чем ранее используемые границы. Пусть решение задачи о назначениях содержит некоторое число циклов, как это показано в качестве примера на рис. 10.20 (а). Пусть i-й цикл обозначается $S_{1,\,i}$, и пусть число циклов равно n_1. (Мы будем использовать один и тот же символ $S_{1,\,i}$ как для обозначения цикла, так и для обозначения множества его вершин.)

Определим *стягивание* как замену цикла единственной вершиной. Полученный граф содержит n_1 вершин $S_{1,i}$ ($i = 1, 2, \ldots, n_1$). В качестве матрицы весов нового графа берется ($n_1 \times n_1$)-матрица $\mathbf{C}_1 \equiv [c_1(S_{1,i},\ S_{1,j})]$ с элементами вида

$$c_1(S_{1,\,i},\ S_{1,\,j}) = \min_{\substack{k_i \in S_{1,\,i} \\ k_j \in S_{1,\,j}}} [f_1(k_i,\ k_j)], \tag{10.19}$$

где $F_1 = [f_1(k_i,\ k_j]$ — окончательная матрица относительных весов, полученная в конце решения задачи венгерским (например) методом (см. гл. 12).

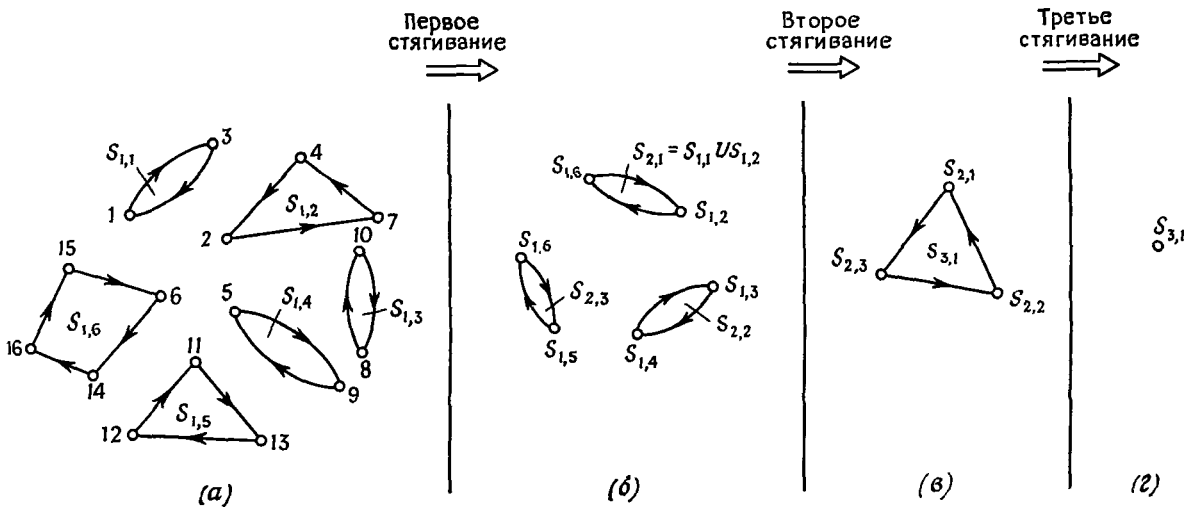


Рис. 10.20. Процесс стягивания. Первое стягивание, второе стягивание, третье стягивание.

На рис. 10.20 (а), например,

$$c_1(S_{1,5}, S_{1,6}) = \min_{\substack{k_5 \in \{11, 12, 13\} \\ k_6 \in \{6, 14, 15, 16\}}} [f_1(k_5, k_6)].$$

Теперь второе решение задачи о назначениях в случае задачи с матрицей $\mathbf{C}_1$ все еще может содержать циклы, в качестве вершин которых выступают предыдущие циклы. Рис. 10.20 (б) показывает один из возможных способов образования новых циклов $S_{2,i}$ ($i = 1, 2, \ldots, n_2$), где n_2 — их общее число (на рис. 10.20 (б) $n_2 = 3$). Эти циклы опять могут быть стянуты в вершины и дать новую задачу с новой матрицей весов $\mathbf{C}_2 \equiv [c_2(S_{2,i}, S_{2,j})]$, вычисленной так же, как и в (10.19), т. е.

$$c_2(S_{2,i}, S_{2,j}) = \min_{\substack{k_i \in S_{2,i} \\ k_j \in S_{2,j}}} [f_2(k_i, k_j)], \qquad (10.20)$$

где множества $S_{2,i}$ являются объединениями всех множеств $S_{1,l}$, образующих «частные» циклы, и где $F_2 = [f_2(k_i, k_j)]$ — матрица относительных весов, полученная в конце решения второй задачи о назначениях.

Решение задачи о назначениях для нового, дважды стянутого, графа все еще может содержать циклы, и итерационный процесс решения — стягивания можно продолжать до тех пор, пока задача не сведется к единственной вершине.

Определим *компрессию* как преобразование матрицы, не удовлетворяющей аксиоме треугольника в метрическом пространстве, в матрицу, удовлетворяющую этой аксиоме. Таким образом, для компрессии матрицы нужно заменить каждый элемент m_{ij}, для которого

$$m_{ij} > m_{ik} + m_{kj} \quad \text{(при некотором } k\text{)},$$

элементом $\min_k [m_{ik} + m_{kj}]$ и продолжать эту замену до тех пор, пока для всякого k не будут справедливы неравенства $m_{ij} \leqslant m_{ik} + m_{kj}$.

Теорема 5. *Сумма значений решений задач о назначениях, полученных во время процесса «решение — стягивание — компрессия» осуществляемого вплоть до момента, когда «стянутая» задача будет содержать единственную точку), является точной нижней границей для задачи коммивояжера.*

Доказательство. В гл. 12 показано, что элемент (i,j) в матрице относительных весов, полученной в конце решения задачи о назначениях, дает нижнюю границу для дополнительного веса, который получается из-за включения дуги (i, j) в решение.

Рассмотрим цикл $S_{1,i}$, полученный в конце решения первой задачи о назначениях. Любой гамильтонов цикл, проходящий

через все n вершин, будет иметь по крайней мере две дуги (одну направленную внутрь цикла $S_{1,i}$, другую — наружу), инцидентные вершинам, не лежащим в $S_{1,i}$, и «приходящие» в одну или более вершин из $S_{1,i}$. Число таких дуг будет, очевидно, четным.

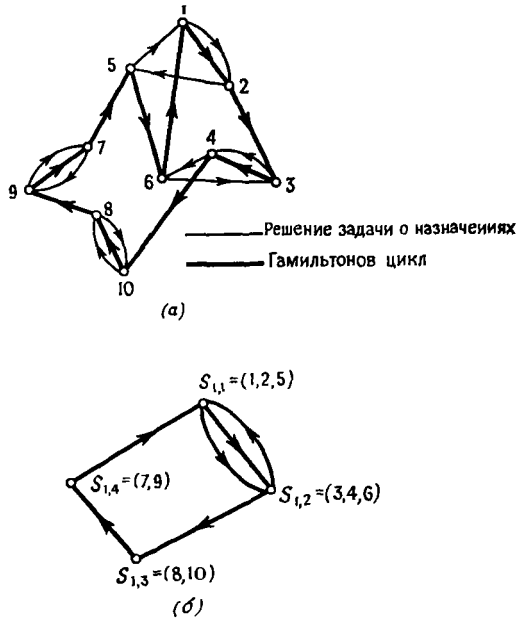


Рис. 10.21. Преобразование гамильтоновых циклов при стягивании. (а) Решение задачи о назначениях. (б) Гамильтонов цикл.

На рис. 10.21 (а) гамильтонов цикл показан жирной линией, в то время как решение задачи о назначениях показано тонкой линией. После стягивания получается граф, изображенный на рис. 10.21 (б), с двумя дугами, инцидентными $S_{1,3}$ и $S_{1,4}$, и четырьмя дугами, инцидентными $S_{1,1}$ и $S_{2,2}$.

Если теперь использовать неравенство треугольника, то получим

$$c_1(S_{1,4}, S_{1,3}) \leqslant c_1(S_{1,4}, S_{1,1}) + c_1(S_{1,1}, S_{1,2}) + c_1(S_{1,2}, S_{1,3}), \quad (10.21)$$

и, следовательно, назначение из рис. 10.22 (в котором дуги $(S_{1,4}, S_{1,1})$, $(S_{1,1}, S_{1,2})$, $(S_{1,2}, S_{1,3})$ заменены дугой $(S_{1,4}, S_{1,3})$) имеет значение, не превосходящее величины назначения из рис. 10.21 (б). Так как назначение на рис. 10.22 имеет по две дуги, инцидентных каждой вершине, и решение задачи о назначениях для стянутого графа имеет меньший вес назначения, то значение решения задачи о назначениях, скажем $V(AP_1)$, является нижней границей для

величины назначения на рис. 10.21 (б), т. е. для значения приращения веса, которое было бы необходимо для объединения вместе различных циклов. Очевидно, что поскольку граф назначения, полученный с помощью любого гамильтонова цикла, содержит

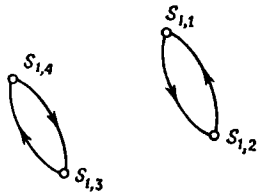


Рис. 10.22. Назначение, соответствующее рис. 10.21 (б), с двумя дугами в каждой вершине.

после первого стягивания (т. е. в случае назначения, соответствующего рис. 10.21 (б)) эйлеров цикл, то его всегда можно преобразовать в граф с меньшим (или равным) весом, в котором каждой вершине индицентны только две дуги. Это делается с помощью замены цепи, образованной дугами, ведущими из одной вершины в другую, единственной дугой между этими вершинами (как было продемонстрировано выше).

Если, с другой стороны, матрица относительных весов $\mathbf{C}_1$ не удовлетворяет условию треугольника, то соотношение (10.21) может быть не выполнено. В этом случае матрица $\mathbf{C}_1$ должна быть сначала подвергнута компрессии. Значение решения задачи о назначениях с компрессированной матрицей следует взять тогда в качестве нижней границы величины приращения веса, что необходимо при объединении циклов. Это так, ибо компрессия матрицы может лишь уменьшить или оставить без изменения вес любого назначения для первоначальной матрицы.

Аналогично $V(AP_2)$ — вес решения задачи о назначениях после второго стягивания является нижней границей веса связываемых циклов, получаемых при этом стягивании, и так далее для третьего, четвертого и остальных стягиваний. Поэтому величина

$$L = \sum_{i=0}^{k} V(AP_i), \tag{10.22}$$

где $V(AP_0)$ — значение начального решения задачи о назначениях с начальной матрицей $\mathbf{C}$, а k — число стягиваний, необходимых для преобразования графа задачи в единственную вершину, равное весу решения задачи коммивояжера. Теорема доказана.

Вообще здесь следует отметить, что даже если начальная матрица весов удовлетворяет условию треугольника, то последующие матрицы относительных весов могут не удовлетворять этому условию и на каком-то этапе может потребоваться компрессия.

Пример вычисления границы

Рассмотрим задачу коммивояжера с 10 вершинами, матрица весов которой симметрична и приведена в табл. 10.3.

Решение начальной задачи о назначениях дает величину $V(AP_0) = 184$. Результирующая матрица относительных весов дана в табл. 10.4, а решение дается графом на рис. 10.23.

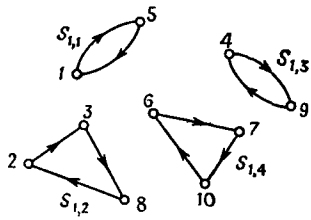


Рис. 10.23. Первое решение задачи о назначениях.

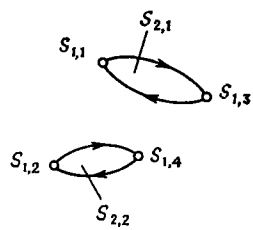


Рис. 10.24. Решение после первого стягивания.

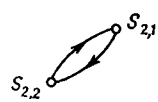


Рис. 10.25. Решение после второго стягивания.

Стягивание графа из рис. 10.23 дает граф с 4 вершинами, матрица весов которого может быть вычислена по (10.19), и она дана в табл. 10.5 (а). Эта матрица весов не удовлетворяет условию треугольника и поэтому подвергается компрессии. Результат дан в табл. 10.5 (б).

Решение задачи о назначениях с матрицей из табл. 10.5 (б) дает величину $V(AP_1) = 20$. Результирующая матрица относительных весов дана в табл 10.6, а решение задачи о назначениях дается графом на рис. 10.24.

Таблица 10.3

Начальная матрица весов

	1	2	3	4	5	6	7	8	9
2	32								
3	41	22							
4	22	50	63						
5	20	42	41	36					
6	57	51	30	78	45				
7	54	61	45	72	36	22			
8	32	20	10	54	32	32	41		
9	22	54	60	20	22	67	57	50	
10	45	51	36	64	28	20	10	32	50

Решение задачи о назначениях →

Таблица 10.4

Матрица относительных весов

	1	2	3	4	5	6	7	8	9	10
1	∞	12	31	2	0	37	44	24	2	37
2	0	∞	0	18	10	19	39	0	22	31
3	19	0	∞	41	19	8	33	0	38	26
4	2	30	53	∞	16	58	62	46	0	56
5	0	22	31	16	∞	25	26	24	2	20
6	25	19	8	46	13	∞	0	12	35	0
7	32	39	33	50	14	0	∞	31	35	0
8	12	0	0	34	12	12	31	∞	30	24
9	2	34	50	0	2	47	47	42	∞	42
10	25	31	26	44	8	0	0	24	30	∞

↓

Таблица 10.5а

Матрица весов

Таблица 10.5б

	1	2	3	4
1	∞	12	2	20
2	0	∞	2	8
3	2	14	∞	22
4	8	8	10	∞

	1	2	3	4
1	∞	12	2	20
2	0	∞	18	8
3	2	30	∞	42
4	8	8	30	∞

Решение задачи о назначениях

Таблица 10.6

	1	2	3	4
1	∞	10	0	10
2	0	∞	2	0
3	0	12	∞	12
4	0	0	2	∞

Таблица 10.7

	1	2
1	∞	0
2	10	∞

Стягивание графа из рис. 10.24 дает граф с 2 вершинами, матрица весов которого (вычисленная по (10.20)) дана в табл. 10.7 (Эта тривиальная (2 × 2)-матрица не должна подвергаться компрессии, так как она удовлетворяет условию треугольника.)

Решение задачи о назначениях с матрицей из табл. 10.7 имеет значение $V(AP_2) = 10$, а само решение дается графом на рис. 10.25, который переходит в единственную вершину после следующего сжатия. Таким образом, нижняя граница значения для задачи коммивояжера с матрицей из табл. 10.3 равна

$$L = V(AP_0) + V(AP_1) + V(AP_4) = 214.$$

Сравнив ее со значением оптимального решения задачи коммивояжера, равного 216, получим ошибку в 0,93 %, в то время как использование в качестве нижней границы величины $V(AP_0)$ дает ошибку в 14,8 %. Хотя настоящая нижняя граница и не является в общем случае такой близкой, как в приведенном примере, но результаты [8] показывают, что эта граница в среднем значительно лучше, чем $V(AP_0)$.

Время вычисления границы в задаче коммивояжера в среднем только на 9% больше времени, требуемого для решения задачи о назначениях с той же самой матрицей весов. Время вычисления при решении задачи о назначениях с использованием венгерского метода оценивается как kn^3 (см. гл. 12), где k — некоторая постоянная, а n — порядок матрицы. Самый плохой случай при вычислении границы (что касается времени вычисления) возникает тогда, когда все циклы при каждом стягивании содержат только по две вершины. В этом случае общее время вычисления при решении задачи о назначениях таково:

$$kn^3 + k\left(\frac{n}{2}\right)^3 + k\left(\frac{n}{4}\right)^3 + \ldots \approx \frac{8}{7}kn^3 \approx 1{,}143\ kn^3. \quad (10.23)$$

Из (10.23) видно, что в худшем случае требуемое время для вычисления предложенной границы в задаче коммивояжера только на 14,3% больше, чем время, требуемое для решения задачи о назначениях того же самого размера. (Здесь следует заметить, что время, необходимое для выполнения той части процесса, которая связана с «стягиванием» и «компрессией», оценивается как n^2.)

7.5. Пример из раздела 7.2 с улучшенной границей

Рассмотрим еще раз пример из раздела 7.2, используя улучшенную границу, описанную выше (вместо использования в качестве такой границы просто значения решения задачи о назначениях). Граница, соответствующая начальной задаче с матрицей весов $\mathbf{C}_A$ (разд. 7.2), должна тогда быть $V(AP_0) + V(AP_1) + 232 + 13 = 245$. (Необходимо второе стягивание, и решение новой задачи о назначениях равно 13.) Подзадачи B, C и D получены, как и раньше,

но новой границей для подзадачи C будет $V(AP_0) + V(AP_1) = 248 + 6 = 254$, вместо предыдущего значения 248. Новой границей для подзадачи D будет $V(AP_0) + V(AP_1) = 250 + 0 = 250$ — то же, что и раньше. (Граница для B останется, конечно, неизменной, так как решение задачи B является на самом деле гамильтоновым циклом. Теперь ветвление нужно делать в узле D (так как $250 < 254$), а не в C, как в предыдущем случае. Это ветвление приводит, как и прежде, к подзадачам G, H и I, и наилучшее решение (подзадача H) имеет значение 251. Дальнейшее ветвление в C (где решение не является гамильтоновым

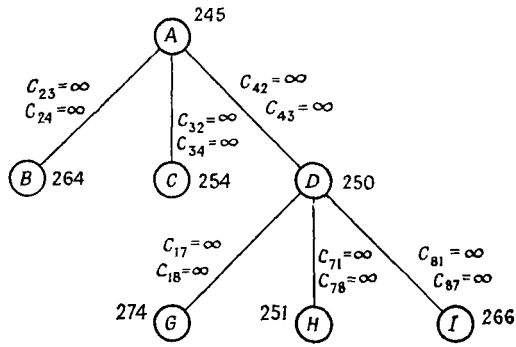


Рис. 10.26. Поиск с деревом решений из примера 7.2 с улучшенной границей для задачи о назначениях.

циклом) уже не будет необходимым, так как его нижняя граница, а именно 254, больше чем 251. Прежняя граница для C, равная 248 (меньше 251), была недостаточной для того, чтобы прекратить ветвление в этом узле. Таким образом, улучшенная граница позволяет сэкономить два узла в дереве решений поиска, как это показано на рис. 10.26. Это дает экономию только на 2/9 для этой маленькой задачи, но для задач большого размера получается большой процент экономии узлов дерева, т. е. числа задач о (n на n) назначениях, которые следует решить

8. Задачи

1. Показать, что если неориентированный граф G удовлетворяет условиям: (1) для каждого положительного целого числа $k <$ $< \frac{1}{2}(n-1)$, число вершин со степенью, не превосходящей k, меньше чем k, (2) если число вершин со степенью, не превосходящей $\frac{1}{2}(n-1)$, меньше или равно $\frac{1}{2}(n-1)$, то оно имеет гамиль-

тонов цикл (см. [25, 29]. Заметим также, что граф, состоящий из единственного гамильтонова цикла, не удовлетворяет вышеприведенным условиям, т. е. условия достаточны, но не являются необходимыми).

2. Доказать, что если в неориентированном графе G для любой пары несмежных вершин x_i и x_j степени удовлетворяют условию

$$d(x_i) + d(x_j) \geqslant n,$$

то граф G имеет гамильтонов цикл (см. [26]).

3. Рассмотрим решетчатый граф G, образованный p горизонтальными и q вертикальными линиями, где каждая точка пересечения рассматривается как вершина и каждый отрезок между соседними точками пересечения решетки — как ребро. При каких p и q граф G имеет гамильтонов цикл.

4. Показать, что граф, вершины и ребра которого соответствуют вершинам и ребрам n-мерного гиперкуба, имеет гамильтонов цикл.

5. Показать, что в графе из задачи 4 гамильтонова цепь между двумя диаметрально противоположными вершинами существует тогда и только тогда, когда n нечетно.

6. Доказать, что в каждом полном антисимметрическом ориентированном графе существует гамильтонова цепь.

7. Используя методы разд. 2.1, 2.2 и 2.3, либо найти все гамильтоновы циклы в графе из рис. 10.27 (если такие циклы существуют), либо показать, что таких циклов нет. Сравнить вычислительную трудоемкость этих методов.

8. Используя метод разд. 2.3, проверить, что граф, изображенный на рис. 10.28, не содержит гамильтонова цикла. Доказать тот же результат другим способом.

9. Для задачи коммивояжера с приводимой ниже матрицей реберных весов **C** вычислить нижние границы оптимальных решений, используя задачи о назначениях и о кратчайшем остове.

$$\mathbf{C} = \begin{array}{c|cccccc} & 1 & & & & & \\ 2 & 41 & 2 & & & & \\ 3 & 12 & 30 & 3 & & & \\ 4 & 27 & 40 & 19 & 4 & & \\ 5 & 52 & 14 & 39 & 43 & 5 & \\ 6 & 62 & 25 & 53 & 65 & 31 & 6 \\ 7 & 47 & 17 & 38 & 52 & 28 & 15 \end{array}$$

10. Найти кратчайшую гамильтонову цепь между вершинами 4 и 7 для графа из задачи 9.

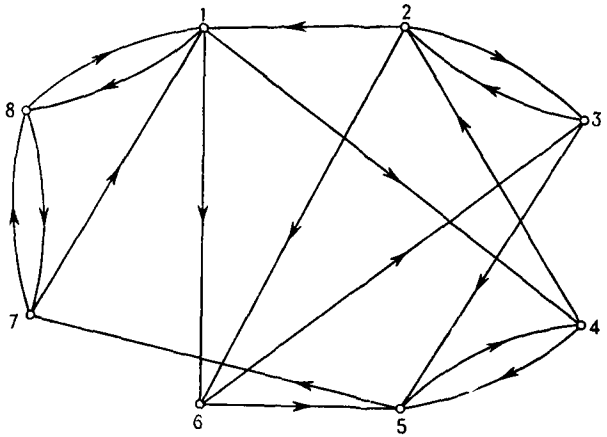


Рис. 10.27.

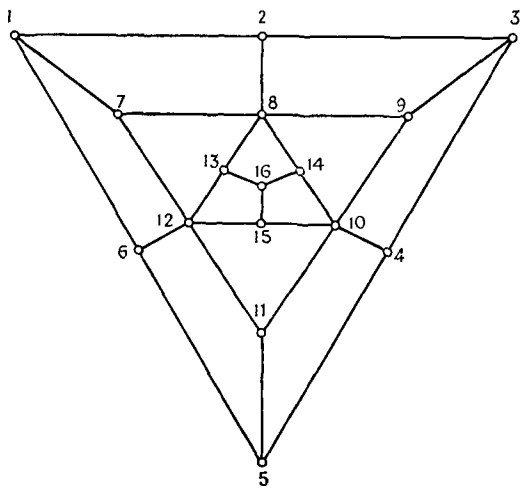


Рис. 10.28.

11. Найти абсолютно кратчайшую гамильтонову цепь в граф из задачи 9.

12. Решить задачу коммивояжера с матрицей реберных весо из задачи 9, используя алгоритм поиска с деревом решений и раздела 7.1 и правило ветвления Б.

13. Решить задачу 12, используя тот же самый метод, но пр вычислении нижних границ, действуя в соответствии с результата ми разд. 7.4.

9. Список литературы

1. Bellmore M., Nemhauser G. L. (1968), The travelling salesman problem — a survey, *Ops. Res.*, **16**, p. 538.
2. Bellmore M., Malone J. C. (1971), Pathology of travelling salesman subtour-elimination algorithms, *Ops. Res.*, **19**, p. 278.
3. Burstall R. M. (1967), Tree-searching methods with an application to a network design problem, Machine Intelligence, Vol. 1, Collins and Michie, Eds., Oliver and Boyd, London.
4. Camerini P. M., Fratta L., Maffioli F. (in press), The travelling salesman problem: heuristically guided search and modified gradient techniques.
5. Charlton J. M., Death C. C. (1970), A method of solution for general machine-scheduling problems, *Ops. Res.*, **18**, p. 689.
6. Christofides N. (1973), Large scheduling problems with bivalent costs, *The Computer J.*, **16**, p. 263.
7. Christofides N. (1970), The shortest Hamiltonian chain of a graph, *J. of SIAM (Appl. Math)*, **19**, p. 689.
8. Christofides N. (1972), Bounds for the travelling salesman problem, *Ops. Res.*, **20**, p. 1044.
9. Christofides N., Eilon S. (1969), An algorithm for the vehicle dispatching problem, *Opl. Res. Quart.*, **20**, p. 309.
10. Christofides N., Eilon S. (1972), Algorithms for large-scale travelling salesman problems, *Opl. Res. Quart.*, **23**, p. 511.
11. Danielson G. H. (1968), On finding the simple paths and circuits in a graph, *IEEE Trans.*, **CT-15**, p. 294.
12. Demoucron G., Malgrange Y., Petruiset R. (1964), Graphes planaires; reconnaissance at construction de representations planaires topologiques, *Rev. Fran. de Resh. Oper.*, **8**, p. 33.
13. Deo N., Hakimi S. L. (1965), The shortest generalized Hamiltonian tree, Proc. 3rd Allerton Conference on Circuit and Systems Theory, University of Illinois, Urbana, p. 879.
14. Dhawan V. (1969), Hamiltonian circuits and related problems in graph theory, M. Sc. Report, Imperial College, London.
15. Eilon S., Watson-Gandy C. D. T., Christofides N. (1971), Distribution Management: Mathematical modelling and practical analysis, Griffin, London.
16. Gilmore P. C., Gomory R. E. (1964), Sequencing a one-state variable machine; a solvable case of the travelling salesman problem, *Ops. Res.*, **12**, p. 655.
17. Haring D. R. 1966), Sequential-circuit synthesis, MIT Press, Research Monograph 31, Cambridge, Massachusetts.
18. Held M., Karp R. M. (1970), The travelling salesman problem and minimum spanning trees, *Ops. Res.*, **18**, p. 1138.
19. Held M., Karp R. M. (1971), The travelling salesman problem and minimum spanning trees. Part II, *Math. Progr.*, **1**, p. 6.
20. Krolak P., Felts W., Marble G. (1971), A man-machine approach toward solving the travelling salesman problem, *Comm. of ACM*, **14**, p. 327.
21. Lawler E. L. (1971), A solvable case of the travelling salesman problem, *Math. Prog.*, **1**, p. 267.
22. Lin S. (1965), Computer solutions of the travelling salesman problem, *Bell Syst. Tech. J.*, 44, p. 2245.
23. Lin S., Kernighan B. W. (1971), A heuristic technique for solving a class of combinatorial optimization problems, Princeton Conf. on System Science.
24. Maffioli F. (1973), The travelling salesman problem and its implications, Report, Instituto di Elettrotecnica ed Elettronica, Politechnico di Milano.

25. Nash-Williams C, St, J. A. (1966), On Hamiltonian circuits in finite graphs, *Proc. American Mathematical Soc.*, **17**, p. 466.
26. Ore O. (1962), Theory of Graphs, American Mathematical Society, New York.
27. Перепелица В. А., Гимади Э. Х. (1969), К задаче нахождения минимального гамильтонова контура на графе со взвешенными дугами, сб. «Дискретный анализ», Новосибирск, стр. 57—65.
28. Pohl J. (1970), Heuristic search viewed as path in a graph, *Artificial Intelligence*, **1**, p. 193.
29. Pósa L. (1962), A theorem concerning Hamiltonian lines, *Magyar Tnd. Akad. Mat. Kutató Int. Közl.*, **7**, p. 225.
30. Roberts S. M., Flores B. (1967), An engineering approach to the travelling salesman problem, *Man. Sci.*, **13**, p. 269.
31. Roberts S. M., Flores B. (1966), Systematic generation of Hamiltonian circuits, *Comm. of ACM*, **9**, p. 690.
32. Roy B. (1959), Recherche des circuits elementaires et des circuits Hamiltoniens dans un graphe quelconque, Mimeographe, Soc. de Math. Appl., Paris.
33. Roy B. (1969, 1970), Algebre modern et theorie des graphes, Vols 1 an 2, Dunod, Paris.
34. Рубинштейн М. И. (1971), О симметричной задаче коммивояжера, *Автоматика и телемеханика*, 9, стр. 126.
35. Selby G. R. (1970), The use of topological methods in computer-aided circuit layout, Ph. D. Thesis, London University.
36. Syslo M. M. (1973), A new solvable case of the travelling salesman problem, *Math. Prog.*, **4**, p. 347.
37. Yau S. S. (1967), Generation of all Hamiltonian circuits, paths and centres of a graph and related prolblems, *IEEE Trans.*, **CT-14**, p. 79.

10. Приложение

Ниже излагается алгоритм вычисления верхней границы B минимального числа перенастроек аппаратуры, требующихся в задаче планирования из разд. 4 данной главы, который можно использовать в качестве начального алгоритма этого раздела.

Шаг 0. Индекс $\leftarrow 0$, $B \leftarrow 0$.

Шаг 1. Взять метки $l(x_i) = 0$, $\forall\, x_i \in X$. Положить $p = 0$.

Шаг 2. $G = (X, A)$, выбрать любое $x_0 \in X$, положить $S = \{x_0\}$.

Шаг 3. Если индекс $=0$, образовать $\bar{S} = S \cup \Gamma(S)$; в противном случае $\bar{S} = S \cup \Gamma^{-1}(S)$. Здесь $\Gamma(x_i) = \{x_j \mid (x_i, x_j) \in A\}$ и $\Gamma(S) = \bigcup\limits_{x_i \in S} \Gamma(x_i)$, а также

$$\Gamma^{-1}(x_i) = \{x_j \mid (x_j, x_i) \in A\} \text{ и } \Gamma^{-1}(S) = \bigcup_{x_i \in S} \Gamma^{-1}(x_i).$$

Шаг 4. Если $\bar{S} = S$, то перейти к шагу 5; в противном случае $p \leftarrow p + 1$, $l(x_i) \leftarrow p\ \forall\, x_i \in \bar{S} - S$, $\bar{S} \leftarrow S$ и вернуться к шагу 3.

Шаг 5. Найти $x \in \{x_i \mid l(x_i) = p\}$.

Шаг 6. Если индекс=0, найти $x' \in \{x_i \mid l(x_i) = p - 1$ и $(x', x) \in A\}$; в противном случае найти $x' \in \{x_i \mid l(x_i) = p - 1$ и $(x, x') \in A\}$.

Шаг 7. $X \leftarrow X - \{x\}$, $A \leftarrow A - \{(x, x_i) \mid x_i \in X\} - \{(x_i, x) \mid x_i \in X\}$. Если $x = \{x_o\}$, то $B \leftarrow B + 1$ и перейти к шагу 12; в противном случае перейти к шагу 8.

Шаг 8. $x \leftarrow x'$, $p \leftarrow p - 1$. Если $x' = x_o$, перейти к шагу 9; в противном случае перейти к шагу 5.

Шаг 9. Если индекс $=0$, индекс $\leftarrow 1$ и перейти к шагу 1; в противном случае перейти к шагу 10.

Шаг 10. Если $X = \{x_o\}$, то $B \leftarrow B + 1$ и перейти к шагу 12; в противном случае перейти к шагу 11.

Шаг 11. Индекс $\leftarrow 0$, $B \leftarrow B + 1$.

$X \leftarrow X - \{x_o\}$, $A \leftarrow A - (\{(x_o, x_i) \mid x_i \in X\} - \{(x_i, x_o) \mid x_i \in X\})$.

Шаг 12. Стоп. B является требуемой верхней границей.

Алгоритм требует некоторых пояснений. Если индекс $=0$, то прослеживаются *прямые* цепи, проходящие по вершинам графа G, начиная с вершины x_o. Эти цепи прослеживаются с присвоением метки p тем вершинам из G, которые могут быть достигнуты из x_o через p дуг (шаги 1, 2, 3 и 4). Если ни одна из этих цепей не может быть продолжена, то алгоритм переходит к шагам 5, 6 и 7, в которых самая длинная из этих цепей прослеживается назад к вершине x_o, при этом убираются из графа вершины (и ассоциированные с ними дуги), лежащие на этой самой длинной цепи. Шаг 8 прекращает процесс удаления вершин. Шаг 9 возвращает к началу алгоритма со значением индекса $=1$, чтобы начать формирование *обратных* цепей, т. е. цепей, оканчивающихся в x_o. Снова находится и удаляется самая длинная из этих цепей.

Самая длинная из прямых и обратных цепей (*в* или *из* x_o) может рассматриваться как одна длинная цепь, содержащая x_o. Число B (необходимых последовательностей цепей) увеличивается тогда на единицу на шаге 11, и после выбора другой вершины x_o из оставшегося графа процесс продолжается (формируются новые длиннейшие прямые и обратные цепи и т. д.), пока не будет исчерпан весь граф.

Конечное значение B, которое равно числу последовательностей цепей, необходимых для «стирания» всего графа (т. е. для покрытия всех вершин), является тогда, очевидно, верхней границей для требуемого минимального числа таких покрывающих последовательностей (т. е. минимального числа перенастроек аппаратуры).

Глава 11

ПОТОКИ В СЕТЯХ

1. Введение

Одной из наиболее интересных и важных задач теории графов является задача определения максимального потока, протекающего от некоторой вершины s графа (источника) к некоторой конечной вершине t (стоку). При этом каждой дуге (x_i, x_j) графа G приписана некоторая пропускная способность q_{ij}, и эта пропускная способность определяет наибольшее значение потока, который может протекать по данной дуге. Эта задача и ее варианты могут возникать во многих практических приложениях, например при определении максимальной интенсивности транспортного потока между двумя пунктами на карте дорог, представляемой графом. В этом примере решение задачи о максимальном потоке укажет также ту часть сети дорог, которая «насыщена» и образует «узкое место» в отношении потока между двумя указанными концевыми пунктами.

Метод решения *задачи о максимальном потоке* (от s к t) был предложен Фордом и Фалкерсоном [12], и их «техника пометок» составляет основу других алгоритмов решения многочисленных задач, являющихся простыми обобщениями или расширениями указанной задачи. Следующие возможные варианты задачи о максимальном потоке (от s к t) встречаются в литературе.

(I) Допустим, что каждой дуге графа приписана не только пропускная способность q_{ij}, дающая верхнюю границу потока через дугу (x_i, x_j), но также «пропускная способность» r_{ij}, дающая нижнюю границу потока через дугу. В таком случае далеко не очевидно, что существует допустимое множество потоков, удовлетворяющих требованиям о максимальной и минимальной пропускных способностях дуг. Однако если (в общем случае) существует много таких потоков и если в дополнение к пропускным способностям заданы также стоимости единицы потока, протекающего по дуге, то возникает *задача нахождения допустимого потока минимальной стоимости* (от вершины s к вершине t).

(II) Рассмотрим случай, когда ищутся *максимальные потоки между каждой парой вершин*. Хотя такая задача может быть решена как $n(n-1)/2$ «отдельных» (от s к t) задач, но это очень трудоемкий процесс. При нахождении кратчайших цепей между каждой парой вершин графа было установлено, что нет необходимости решать каждую (от s к t) задачу о кратчайшей цепи индивидуально

(см. гл. 8), и в настоящей задаче точно так же существует метод, который — для неориентированных графов — не предполагает обязательного решения задач о максимальных потоках для каждой пары вершин s и t.

(III) Если вместо одного источника и одного стока рассмотреть несколько заданных источников и стоков, причем между различными источниками и стоками протекают потоки различных продуктов, то задача максимизации суммы всех потоков между источниками и стоками называется задачей о *многопродуктовом потоке*. В этой задаче пропускная способность q_{ij} дуги (x_i, x_j) является ограничением для суммы всех потоков всех видов продуктов через эту дугу.

(IV) Во всех рассмотренных выше случаях неявно допускалось, что поток на входе дуги такой же, как и на выходе. Если отказаться от этого допущения и рассмотреть граф, в котором выходной поток дуги равен ее входному потоку, умноженному на некоторое неотрицательное число, то задачу о максимальном потоке (от s к t) называют задачей о *потоках с выигрышами*. В такой задаче потоки могут и «порождаться», и «поглощаться» самим графом, так что поток, входящий в s, и поток, покидающий t, могут изменяться совершенно независимо.

Область теории графов, в которой рассматриваются вопросы о потоках в графах, интенсивно развивалась в большом числе работ. В настоящей главе дается обзор общих проблем, устанавливается связь между ними и приводится описание некоторых важных алгоритмов, используемых при решении задач о потоках. Последнее необходимо, во-первых, из-за собственной значимости этих алгоритмов, во-вторых, потому, что они позволяют вскрыть такие связи между задачами, которые не очевидны с других точек зрения, и, в-третьих, потому, что алгоритмы, которые будут описаны, могут быть использованы (непосредственно или как элементарные части более сложных алгоритмов) для анализа широкого круга других задач.

В этой главе мы обсудим основную задачу о максимальном потоке (от s к t) и ее обобщения (I), (III) и (IV). Так как алгоритмы для многопродуктового потока имеют совсем другую природу и совсем не так эффективны, как метод пометок, обсуждаемый в этой главе, то мы не будем заниматься здесь этой задачей и отсылаем заинтересованного читателя к [18, 27, 26, 25, 24, 15].

2. Основная задача о максимальном потоке (от s к t)

Рассмотрим граф $G = (X, A)$ с пропускными способностями дуг q_{ij}, источником s и стоком t; $s, t \in X$. Множество чисел ξ_{ij}, определенных на дугах $(x_i, x_j) \in A$, называют потоками в дугах,

если выполняются следующие условия: [1])

$$\sum_{x_j \in \Gamma(x_i)} \xi_{ij} - \sum_{x_k \in \Gamma^{-1}(x_i)} \xi_{ki} = \begin{cases} v, & \text{если} \quad x_i = s, \\ -v, & \text{если} \quad x_i = t, \\ 0, & \text{если} \quad x_i \neq s, t, \end{cases} \tag{11.1}$$

и

$$\xi_{ij} \leqslant q_{ij} \quad \text{для всех} \quad (x_i, x_j) \in A. \tag{11.2}$$

Уравнение (11.1) является уравнением сохранения потока. Оно утверждает, что поток, втекающий в вершину, равен потоку, вытекающему из вершины, за исключением вершин, являющихся источником и стоком (вершин s и t), для которых существуют сетевые вытекания и приток *величины* v соответственно. Соотношения (11.2) указывают просто на то, что пропускные способности ограничены для каждой дуги графа G. Задача состоит в нахождении такого множества потоков по дугам, чтобы величина

$$v = \sum_{x_j \in \Gamma(s)} \xi_{sj} = \sum_{x_k \in \Gamma^{-1}(t)} \xi_{kt} \tag{11.3}$$

была максимальной. Здесь ξ_{sj} и ξ_{kt} записаны для потоков из вершины s в x_j и из x_k в t соответственно.

Алгоритм расстановки пометок, предложенный Фордом и Фалкерсоном [12] для решения этой задачи, основан на следующей теореме (определение понятия разреза см. в гл. 9).

Теорема 1. (Теорема о максимальном потоке и минимальном разрезе [8, 10, 11].) *Величина максимального потока из s в t равна величине минимального разреза $(X_m \rightarrow \bar{X}_m)$, отделяющего s от t.*

Разрез $X_0 \rightarrow \bar{X}_0$ отделяет s от t, если $s \in X_0$ и $t \in \bar{X}_0$. Величиной [2]) такого разреза называется сумма пропускных способностей всех дуг из G, начальные вершины которых лежат в X_0, а конечные в $\bar{X}_0$, т.е.

$$v(X_0 \rightarrow \bar{X}_0) = \sum_{(x_i, x_j) \in (X_0 \rightarrow \bar{X}_0)} q_{ij}.$$

Минимальный разрез $(X_m \rightarrow \bar{X}_m)$ — это разрез с наименьшим таким значением.

Доказательство. Здесь дано конструктивное доказательство теоремы о максимальном потоке и минимальном разрезе, и используемый метод непосредственно приводит к алгоритму расстановки пометок, излагаемому ниже.

---

[1]) Если могут возникнуть неясности, то мы будем употреблять запись $\xi(x_i, x_j)$, $q(x_i, x_j)$, $r(x_i, x_j)$ и (x_i, x_j) вместо сокращенных обозначений ξ_{ij}, q_{ij}, r_{ij} и c_{ij} соответственно.

[2]) Или *пропускной способностью.— Прим. ред.*

Совершенно очевидно, что максимальный поток из s в t не может быть больше, чем $v(X_m \rightarrow \widetilde{X}_m)$, так как всякая цепь, ведущая из s в t, проходит хотя бы через одну дугу данного разреза. Поэтому достаточно установить существование потока с таким значением. Допустим теперь, что поток задается m-мерным вектором ξ, и определим разрез $(X_o \rightarrow \widetilde{X}_o)$ рекурсивным применением нижеуказанного шага (б).

(а) Начать, полагая $X_o \leftarrow \{s\}$.

(б) Если $x_i \in X_o$ и, кроме того, $\xi_{ij} < q_{ij}$ или $\xi_{ji} > 0$, то включить x_j в множество X_o, и повторять этот шаг до тех пор, пока множество X_o нельзя будет расширять дальше.

Теперь могут возникнуть два случая в зависимости от того, будет ли $t \notin X_o$ или $t \notin X_o$.

Случай (I), $t \in X_o$. В соответствии с шагом (б) отношение $t \in X_o$ означает следующее: существует такая «неориентированная» цепь, ведущая из вершины s в вершину t, что для каждой дуги (x_i, x_j), используемой цепью в прямом направлении (прямой дуги), $\xi_{ij} < q_{ij}$, а для каждой дуги (x_k, x_l), используемой цепью в обратном направлении, т. е. в направлении от x_l к x_k (обратной дуги), $\xi_{kl} > 0$. (Такая цепь (из дуг), ведущая из s в t, будет называться *аугментальной* [1]) *цепью потока.*)

Пусть

$$\delta_f = \min_{(x_i, x_j)} [q_{ij} - \xi_{ij}] \text{ для прямой дуги } (x_i, x_j), \tag{11.5}$$

$$\delta_b = \min_{(x_k, x_l)} [\xi_{kl}] \text{ для обратной дуги } (x_k, x_l), \tag{11.6}$$

и

$$\delta = \min [\delta_f, \delta_b]. \tag{11.7}$$

Если теперь величину δ прибавить к потоку во всех прямых дугах и вычесть во всех обратных дугах цепи, то в результате получится новый допустимый поток, на δ единиц больший, чем предыдущий. Это очевидно в силу того, что прибавление величины δ к потоку в прямых дугах не может привести к превышению ни одной из пропускных способностей этих дуг (так как $\delta \leqslant \delta_f$). а вычитание δ из потока в обратных дугах не может сделать поток в этих дугах отрицательным (так как $\delta \leqslant \delta_b$).

Используя новый исправленный поток, можно затем повторить шаги (а) и (б) и, определив новый разрез $(X_o, \widetilde{X}_o)$, повторить предыдущие рассуждения.

Случай (II), $t \notin X_o$ (т. е. $t \in \widetilde{X}_o$). В соответствии с шагом (б) $\xi_{ij} = q_{ij}$ для всех $(x_i, x_j) \in (X_o \rightarrow \widetilde{X}_o)$ и $\xi_{kl} = 0$ для всех $(x_k, x_l) \in$

[1]) От augmen (*лат.*) — увеличение, приращение, прирост.— *Прим. перев.*

$\in (\widetilde{X}_0 \to X_0)$. Следовательно,

$$\sum_{(x_i, x_j)\in(X_0\to\widetilde{X}_0)} \xi_{ij} = \sum_{(x_i, x_j)\in(X_0\to\widetilde{X}_0)} q_{ij}$$

и

$$\sum_{(x_k, x_l)\in(\widetilde{X}_0\to X_0)} \xi_{kl} = 0,$$

т. е. величина потока, а именно

$$\sum_{(x_i, x_j)\in(X_0\to\widetilde{X}_0)} \xi_{ij} - \sum_{(x_k, x_l)\in(\widetilde{X}_0\to X_0)} \xi_{kl},$$

равна величине разреза $(X_0 \to \widetilde{X}_0)$.

Так как в случае (I) поток все время увеличивается по крайней мере на единицу, то при целочисленности всех q_{ij} максимальный поток получим за конечное число шагов — как только возникнет случай (II). Этот поток будет равен тогда величине текущего разреза $(X_0 \to \widetilde{X}_0)$, который, следовательно, должен тогда быть равным минимальному разрезу. Теорема доказана.

Конструктивный метод, использованный при доказательстве теоремы о максимальном потоке и минимальном разрезе, позволяет сразу же получить алгоритм вычисления максимального потока из данной вершины s в данную вершину t в графе с известными пропускными способностями. Такой алгоритм будет сейчас описан.

Алгоритм начинает работу с произвольного допустимого потока (можно взять и нулевой поток), затем стремятся увеличить величину потока с помощью систематического поиска всех возможных аугментальных цепей потока от s к t. Поиск аугментальной цепи осуществляется с помощью расстановки пометок в вершинах графа. Пометки указывают, вдоль каких дуг может быть увеличен поток и на сколько. Как только найдена одна из таких цепей, поток вдоль нее увеличивают до максимального значения, все пометки в вершинах стираются и вновь полученный поток используется в качестве исходного при новой расстановке пометок. Алгоритм заканчивает работу и дает максимальный поток, если нельзя найти ни одну аугментальную цепь. Алгоритм применяется следующим образом.

2.1. Алгоритм расстановки пометок для задачи о максимальном (от s к t) потоке

А. Расстановка пометок. Вершина может находиться в одном из трех состояний: вершине приписана пометка и вершина просмотрена (т. е. она имеет пометку и все смежные с ней вершины

«обработаны»), пометка приписана, но вершина не просмотрена (т. е. она имеет пометку, но не все смежные с ней вершины обработаны), вершина не имеет пометки. Пометка произвольной вершины x_i состоит из двух частей и имеет один из двух видов: $(+x_j, \delta)$ или $(-x_j, \delta)$. Часть $+x_j$ пометки первого типа означает, что поток допускает увеличение вдоль дуги (x_j, x_i). Часть $-x_j$ пометки другого типа означает, что поток может быть уменьшен вдоль дуги (x_i, x_j). В обоих случаях δ задает максимальную величину дополнительного потока, который может протекать от s к x_i вдоль построенной аугментальной цепи потока. Присвоение пометки вершине x_i соответствует нахождению аугментальной цепи потока от s к x_i. Сначала все вершины не имеют пометок.

Шаг 1. Присвоить вершине s пометку $(+s, \delta(s) = \infty)$. Вершине s присвоена пометка и она просмотрена, все остальные вершины без пометок.

Шаг 2. Взять некоторую непросмотренную вершину с пометкой; пусть ее пометка будет $(\pm x_k, \delta(x_i))$.

(I) Каждой непомеченной вершине $x_j \in \Gamma(x_i)$, для которой $\xi_{ij} < q_{ij}$, присвоить пометку $(-x_i, \delta(x_j))$, где

$$\delta(x_j) = \min[\delta(x_i), q_{ij} - \xi_{ij}].$$

(II) Каждой непомеченной вершине $x_j \in \Gamma^{-1}(x)$, для которой $\xi_{ji} > 0$, присвоить пометку $(-x_i, \delta(x_j))$, где

$$\delta(x_j) = \min[\delta(x_i), \xi_{ji}].$$

(Теперь вершина x_i и помечена, и просмотрена, а вершины x_j, пометки которым присвоены в (I) и (II), являются непросмотренными.) Обозначить каким-либо способом, что вершина x_i просмотрена.

Шаг 3. Повторять шаг 2 до тех пор, пока либо вершина t будет помечена, и тогда перейти к шагу 4, либо t будет не помечена и никаких других пометок нельзя будет расставить; в этом случае алгоритм заканчивает работу с максимальным вектором потока ξ. Здесь следует отметить, что если X_o — множество помеченных вершин, а $\widetilde{X}_o$ — множество не помеченных, то $(X_o \to \widetilde{X}_o)$ является минимальным разрезом.

Б. Увеличение потока

Шаг 4. Положить $x = t$ и перейти к шагу 5.

Шаг 5. (I) Если пометка в вершине x имеет вид $(+z, \delta(x))$, то изменить поток вдоль дуги (z, x) с $\xi(z, x)$ на $\xi(z, x) + \delta(t)$.

(II) Если пометка в вершине x имеет вид $(-z, \delta(x))$, то изменить поток вдоль дуги (x, z) с $\xi(x, z)$ на $\xi(x, z) - \delta(t)$.

Шаг 6. Если $z = s$, то стереть все пометки и вернуться к шагу 1, чтобы вновь начать расставлять пометки, но используя уже улучшенный поток, найденный на шаге 5. Если $z \neq s$, то взять $x = z$ и вернуться к шагу 5.

2.2. Пример

Рассмотрим граф, изображенный на рис. 11.1, и возьмем в качестве источника вершину x_1, а в качестве стока — вершину x_9. Пропускные способности дуг указаны на рисунке. Требуется найти максимальный поток от x_1 к x_9.

В качестве начального возьмем поток с нулевыми значениями на всех дугах. Алгоритм работает следующим образом.

Шаг 1. Припишем вершине x_1 пометку $(+x_1, \infty)$.

Шаг 2. (I) Множество вершин $\{x_j \mid x_j \in \Gamma(x_1), \xi_{ij} < q_{ij}, x_j$ не помечена$\}$ есть $\{x_2, x_4\}$,

вершине x_2 приписывается пометка $(+x_1, \min[\infty, 14 - 0]$, т. е. $(+x_1, 14)$,

вершине x_4 приписывается $(+x_1, \min[\infty, 23 - 0])$, т. е. $(+x_1, 23)$.

(II) Множество вершин $\{x_j \mid x_j \in \Gamma^{-1}(x_1), \xi_{j1} > 0, x_j$ не помечена$\}$ является пустым. Итак, x_1 помечена и просмотрена, x_2 и x_4 помечены и не просмотрены, а все остальные вершины не помечены. Повторяем шаг 2 и первой просматриваем вершину x_2.

(I) Множество $\{x_j \mid x_j \in \Gamma(x_2), \xi_{2j} < q_{2j}$ и x_j не помечена$\}$ есть $\{x_3\}$.

для x_3 пометкой будет $(+x_2, \min[14, 10 - 0] = (+x_2, 10)$.

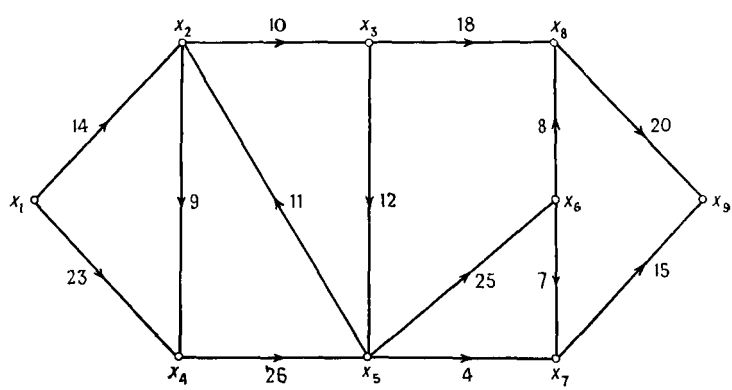


Рис. 11.1. Граф из примера 2.2.

(II) Множество $\{x_j \mid x_j \in \Gamma^{-1}(x_2),\ \xi_{j2} < 0$ и x_j не помечена$\}$ является пустым. Теперь вершины x_1 и x_2 помечены и просмотрены, а x_3 и x_4 помечены, но не просмотрены.

Беря для просмотра x_3 и повторяя шаг 2, придем к следующим пометкам:

для x_5 пометкой будет $(+x_3,\ \min[10,\ 12 - 0]) = (+x_3,\ 10)$,

для x_8 пометкой будет $(+x_3,\ \min[10,\ 18 - 0]) = (+x_3,\ 10)$.

Беря для просмотра x_4, найдем, что никаких пометок расставить нельзя. Продолжая просмотр с x_5, получим следующие пометки:

пометка для x_6 будет $(+x_5,\ \min[10,\ 25 - 0]) = (+x_5,\ 10)$,

пометка для x_7 будет $(+x_5,\ \min[10,\ 4 - 0]) = (+x_5,\ 4)$,

пометка для x_9 будет $(+x_7,\ \min[4,\ 15 - 0]) = (+x_4,\ 4)$.

Переходя к шагам 4 и 5, получим:

$$x = x_9; \qquad \xi_{7,9} = 0 + 4 = 4;$$
$$x = x_7; \qquad \xi_{5,7} = 4;$$
$$x = x_5; \qquad \xi_{3,5} = 4;$$
$$x = x_3; \qquad \xi_{2,3} = 4;$$
$$x = x_2; \qquad \xi_{1,2} = 4.$$

Вид потока в конце шага 5 и пометки вершин до их стирания на шаге 6 показаны на рис. 11.2а. Все потоки показаны подчеркиванием.

Стирая пометки у вершин и возвращаясь к шагу 1 для второго прохода, получим следующие новые пометки вершин (помеченные, но не просмотренные вершины просматриваются в порядке возрастания их номеров).

Шаг 3.

Пометкой для x_1 будет $(+x_1,\ \infty)$,
пометкой для x_2 будет $(+x_1,\ \min[\infty,\ 14 - 4]) = (+x_1,\ 10)$,
пометкой для x_4 будет $(+x_1,\ \min[\infty,\ 23 - 0]) = (+x_1,\ 23)$;
теперь вершина x_1 помечена и просмотрена.
Пометкой для x_3 будет $(+x_2,\ \min[10,\ 10 - 4]) = (+x_2,\ 6)$,
теперь вершина x_2 помечена и просмотрена.
Пометкой для x_5 будет $(+x_3,\ \min[6,\ 12 - 4]) = (+x_3,\ 6)$,
пометкой для x_8 будет $(+x_3,\ \min[6,\ 18 - 0]) = (+x_3,\ 6)$;
теперь вершина x_3 помечена и просмотрена,
вершина x_4 также помечена и просмотрена.

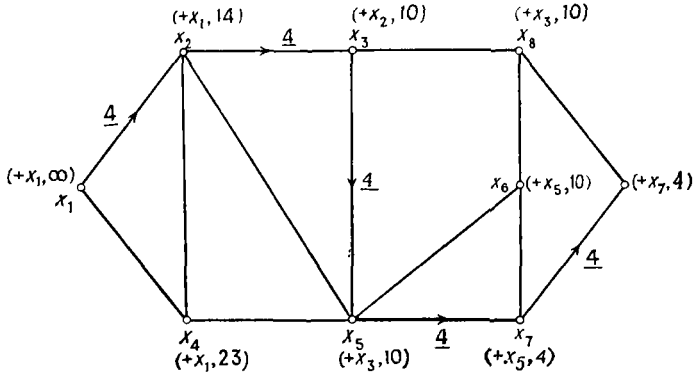


Рис. 11.2а. Потоки и пометки после 1-й итерации.
— насыщенные дуги.

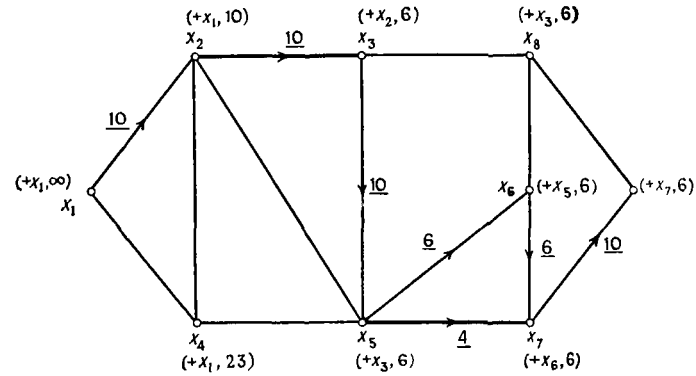


Рис. 11.2б. Потоки и пометки после 2-й итерации.
— насыщенные дуги.

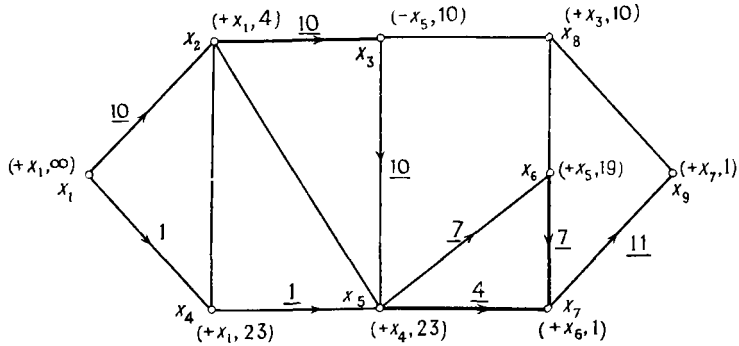


Рис. 11.2в. Потоки и пометки после 3-й итерации.
— насыщенные дуги.

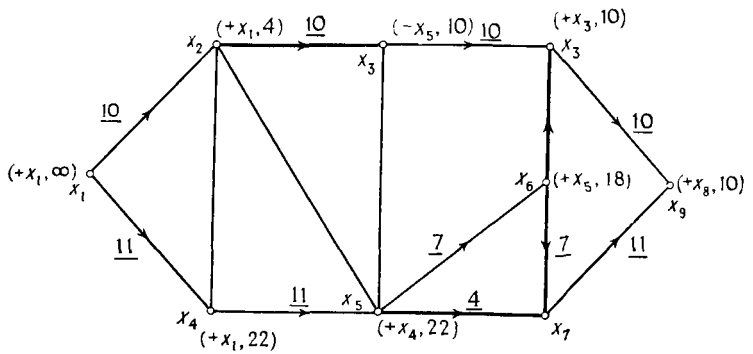


Рис. 11.2г. Потоки и пометки после 4-й итерации.
— насыщенные дуги.

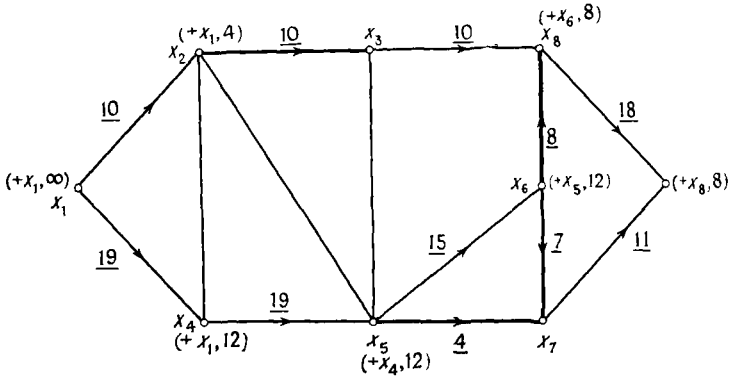


Рис. 11.2д. Потоки и пометки после 5-й итерации.
— насыщенные дуги.

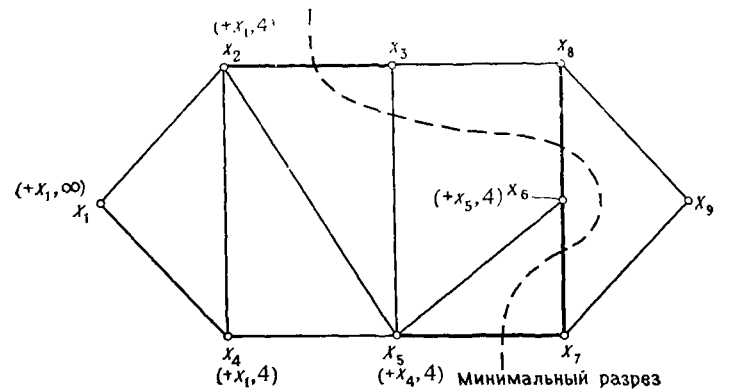


Рис. 11.2е. Потоки и пометки после 6-й итерации.
— насыщенные дуги.

Пометка для x_6 будет $(+x_5, \min[6{,}25 - 0] = (+x_5, 6)$, теперь вершина x_5 помечена и просмотрена.
Пометкой для x_7 будет $(+x_6, \min[6{,}7 - 0]) = (+x_6, 6)$, теперь вершина x_6 помечена и просмотрена.
Пометкой для x_9 будет $(+x_7, \min[6{,}15 - 4]) = (+x_7, 6)$.

Шаги 4 и 5.

Новые потоки увеличились следующим образом:

$$\xi_{7,9} = 4 + 6 = 10; \quad \xi_{6,7} = 0 + 6 = 6; \quad \xi_{5,6} = 0 + 6 = 6;$$
$$\xi_{3,5} = 4 + 6 = 10; \quad \xi_{2,3} = 4 + 6 = 10; \quad \xi_{1,2} = 4 + 6 = 10;$$

все остальные значения потока не изменились.

Новый вид потока и пометки вершин до стирания показаны на рис. 11.2б.

Продолжая дальше, получаем после каждого прохода алгоритма потоки и пометки, изображенные последовательно на рисунках 11.2в—11.2д. Алгоритм заканчивает работу, когда вершина не может быть помечена; «заключительные» пометки показаны на рис. 11.2е.

Поток на рис. 11.2е является поэтому максимальным потоком со значением 29, а соответствующий минимальный разрез показан пунктиром на рис. 11.2е.

2.3. Инкрементальные графы

Процесс нахождения в графе $G = (X, A)$ аугментальной цепи потока, когда поток по дугам задается вектором ξ, можно рассматривать как процесс нахождения цепи (от s к t) в *инкрементальном* графе $G^\mu(\xi) = (X^\mu, A^\mu)$, определяемом следующим образом:

$$X^\mu = X, \qquad A^\mu = A_1^\mu \cup A_2^\mu,$$

где

$$A_1^\mu = \{(x_i^\mu, x_j^\mu) \mid \xi_{ij} < q_{ij}\},$$

причем пропускная способность дуги $(x_i^\mu, x_j^\mu) \in A_1^\mu$ равна $q_{ij}^\mu = q_{ij} - \xi_{ij}$, и

$$A_2^\mu = \{(x_j^\mu, x_i^\mu) \mid \xi_{ij} > 0\},$$

причем пропускная способность дуги $(x_j^\mu, x_i^\mu) \in A_1^\mu$ равна $q_{ji}^\mu = \xi_{ij}$.

Процедура расстановки пометок в алгоритме, описанном в разд. 2.1, является теперь не чем иным, как методом построения

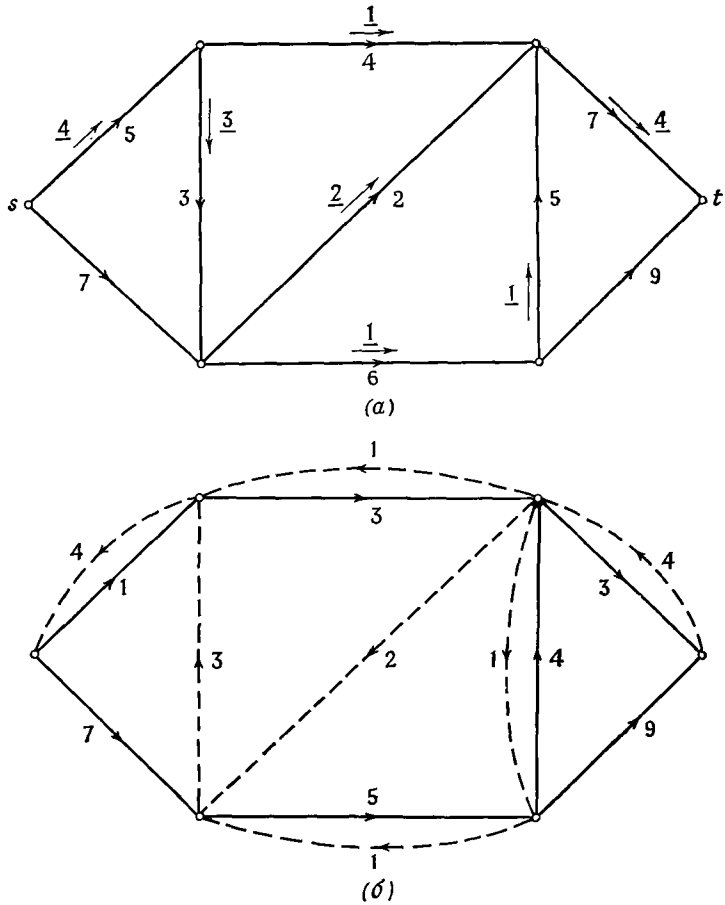


Рис. 11.3. (а) Граф G, потоки ξ_{ij} которого подчеркнуты. (б) Инкрементальный граф $G^{\mu}(\xi)$.

достижимого множества $R(s)$ в инкрементальном графе $G^{\mu}(\xi)$. Если $t \in R(s)$, т. е. если вершина t получила пометку, то в графе $G^{\mu}(\xi)$ найдена цепь P от s к t. Аугментальная цепь в графе G является теперь цепью P, где дуги из P в A_1^{μ} являются прямыми дугами, а дуги из P в A_2^{μ} являются обратными дугами.

На рис. 11.3(а) дан пример вектора ξ в графе, где подчеркнутые числа задают потоки, а другие числа у дуг задают пропускные способности. На рис. 11.3(б) изображен соответствующий инкрементальный граф $G^{\mu}(\xi)$.

2.4. Декомпозиция потока

Иногда бывает желательно представить сложный поток как сумму простых потоков. Это бывает полезно не столько из-за того, что такая декомпозиция нужна на практике, а потому что она дает лучшее понимание природы потоков в сетях и служит удобным средством для обоснования многих потоковых алгоритмов.

Обозначим через $h \circ (S)$ такой поток в графе G, в котором положено $\xi_{ij} = h$ для дуг $(x_i, x_j) \in S$ и $\xi_{ij} = 0$ для дуг $(x_i, x_j) \notin S$. Очевидно, что $h \circ (S)$ не всегда будет потоком, если множество дуг S произвольно, так как поток должен удовлетворять уравнению непрерывности (11.1) при некотором значении v. Совершенно очевидно, что для того, чтобы $h \circ (S)$ представляло поток, множество дуг S должно быть или цепью от s к t в G, или циклом в G.

Для двух заданных потоков ξ и ψ через $\xi + \psi$ обозначим поток, значение которого на дуге (x_i, x_j) равно $\xi_{ij} + \psi_{ij}$.

Теорема 2. *Если ξ — произвольный поток (от s к t) в графе G с (целочисленным) значением v, то ξ можно представить в виде*

$$\xi = 1 \circ (P_1) + 1 \circ (P_2) + \ldots + 1 \circ (P_v) + 1 \circ (\Phi_1) + \\ + 1 \circ (\Phi_2) + \ldots + 1 \circ (\Phi_k),$$

где $P_1, \ldots, P_v$ — простые цепи (от s к t) в графе G, а $\Phi_1, \ldots, \Phi_k$ — простые циклы в G.
(P_i и Φ_i не обязательно должны быть различными).

Доказательство. Для графа $G = (X, A)$ с потоком ξ построим *унитарный* граф $G^e = (X^e, A^e)$ следующим образом. Множество вершин X^e графа G^e совпадает с множеством вершин X графа G. Если ξ_{ij} — поток по дуге (x_i, x_j) графа G, то между вершинами x_i^e и x_j^e графа G^e проведем ξ_{ij} параллельных дуг. Граф G^e будет тогда s-графом, и так как каждая дуга в G^e соответствует единице потока по дуге в G, то граф G^e представляет поток ξ в G.

В графе G^e степени вершин должны удовлетворять (в силу условия непрерывности (11.1)) условиям

$$d_o(x_i^e) = d_t(x_i^e), \qquad \forall x_i^e \neq s^e \text{ или } t^e,$$

$$d_o(s^e) = d_t(t^e) = v.$$

Если теперь добавить к графу G^e v дуг возврата от вершины t^e к вершине s^e, то в графе G^e будет существовать эйлеров цикл (см. гл. 9). Удаление этих v дуг из эйлерова цикла даст v цепей от s^e к t^e, которые в совокупности проходят по каждой дуге графа G^e точно один раз. Пусть эти цепи будут $P_1', P_2', \ldots, P_v'$. Цепи P_i' не обязательно должны быть простыми, хотя (в силу определения эйлерова цикла) в них не должны содержаться одинаковые дуги.

Но так как любую непростую цепь можно рассматривать как сумму простой цепи (от s_e к t_e) и некоторого числа простых попарно непересекающихся по дугам циклов, то

$$\xi = 1 \circ (P_1) + 1 \circ (P_2) + \ldots + 1 \circ (P_v) + 1 \circ (\Phi_1) + \ldots + 1 \circ (\Phi_k),$$

где P_i — простая цепь (от s_e к t_e), а Φ_i — простой цикл. Теорема доказана.

В общем случае не все циклы и цепи будут различными. Если только первые v' цепей и k' циклов различны, причем цепь P_i встречается в списке $P_1, \ldots, P_v$ h_i раз, а цикл Φ_i в списке

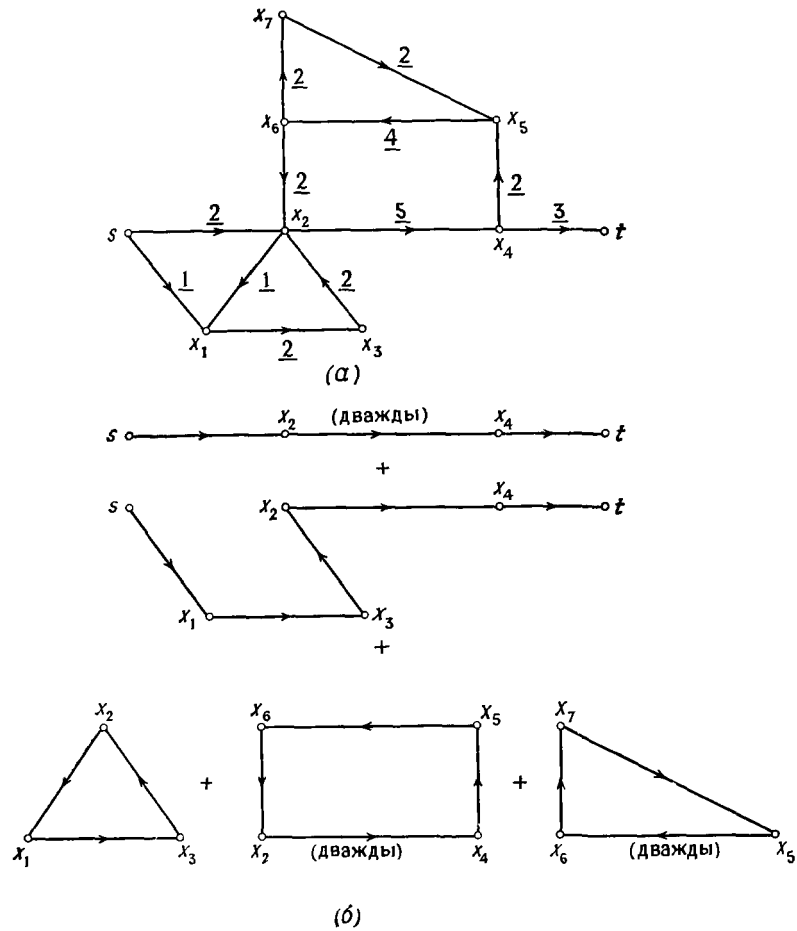


Рис. 11.4. Декомпозиция потока на элементарные (от s к t) потоки по цепям и циклам. (а) Поток ξ. (б) Цепи и циклы потока ξ.

$\Phi_1, \ldots, \Phi_k$, l_i раз, то ξ можно записать в виде

$$\xi = \sum_{i=1}^{v'} h_i \circ (P_i) + \sum_{i=1}^{k'} l_i \circ (\Phi_i),$$

где суммирование понимается как сложение потоков в указанном выше смысле.

В процессе доказательства теоремы 2 получен другой важный результат, а именно что единичные потоки, на которые разложен поток ξ, являются *конформальными*, т. е. если ξ_{ij} — значение потока на дуге (x_i, x_j) графа G, то существует всего ξ_{ij} единичных потоков, которые все используют эту дугу в направлении от x_i к x_j [1]).

Рис. 11.4 дает пример потока ξ и его декомпозицию на простые цепи (от s к t) и циклы.

3. Простые варианты задачи о максимальном потоке (от s к t)

Мы дадим теперь ряд задач, связанных с потоками, которые просто сводятся к задаче о максимальном потоке (от s к t), рассмотренной выше.

3.1. Графы со многими источниками и стоками

Рассмотрим граф с n_s источниками и n_t стоками и предположим, что поток может идти от любого источника к любому стоку. Задачу нахождения максимального общего потока от всех источников ко всем стокам можно преобразовать в простую задачу о максимальном потоке (от s к t) путем добавления нового искусственного источника s и нового искусственного стока t с добавлением дуг, ведущих от s к каждому истинному источнику и от каждого истинного стока к t.

На рис. 11.5 показано, как множество источников и стоков может быть сведено к единственному источнику и единственному стоку. Пропускные способности дуг, ведущих от s к источникам, могут быть выбраны равными бесконечности или в случае, когда производительность источника s_k ограничена, пропускная способность соответствующей дуги (s, s_k) может быть выбрана равной этой границе. Точно так же пропускные способности дуг, ведущих

[1]) Поток не будет конформальным, если, например, существует $\xi_{ij} + p$ единичных потоков, использующих дугу в направлении от x_i к x_j, и p единичных потоков, использующих дугу в противоположном направлеиии, и все эти потоки, таким образом, дают общий поток от x_i к x_j, равный ξ_{ij}. В общем случае два потока ξ и ψ будут конформальными, если для любой дуги (x_i, x_j) будет $\xi_{ij} \cdot \psi_{ji} = 0$.

от стоков к t, могут быть ограничены некоторой величиной (зависящей от стоков) или положены равными бесконечности, если такой границы нет.

Если в задаче некоторые стоки должны снабжаться только определенными источниками и наоборот, то она уже не будет

Рис. 11.5.

простой разновидностью задачи о максимальном потоке (от s к t) и становится задачей о многопродуктовом потоке. Эта задача упоминалась во введении (см. задачу (III)).

3.2. Графы с пропускными способностями дуг и вершин

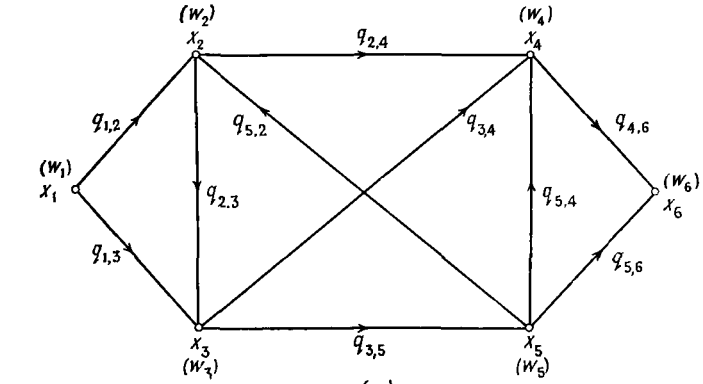
Пусть в графе G дуги имеют пропускные способности q_{ij}, и пусть, в дополнение к этому, вершины графа имеют пропускные способности w_j ($j = 1, 2, \ldots, n$), скажем, такие, что полный поток, входящий в вершину x_j, должен иметь значение, меньшее, чем w_j, т. е.

$$\sum_{x_i \in \Gamma^{-1}(x_j)} \xi_{ij} \leqslant w_j \quad \text{для всех } x_j.$$

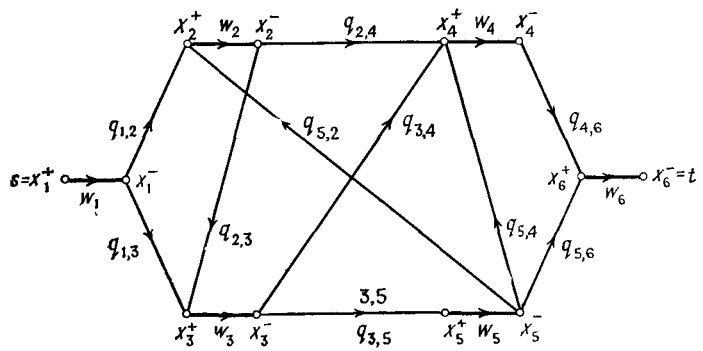
Пусть требуется найти максимальный поток между вершинами s и t такого графа.

Определим граф G_0 так, чтобы каждая вершина x_j графа G соответствовала двум вершинам x_j^+ и x_j^- в графе G_0, причем каждой дуге (x_i, x_j) из G, инцидентной вершине x_j, соответствует дуга (x_i^-, x_j^+) из G_0, инцидентная вершине x_j^+, и каждой дуге (x_j, x_k) из G, исходящей из x_j, соответствует дуга (x_j^-, x_k^+) из G_0, исходящая из x_j^-. Кроме того, введем дугу между x_j^+ и x_j^- с пропускной способностью w_j, т. е. равной пропускной способности вершины x_j.

На рис. 11.6 (а) дан пример графа с пропускными способностями дуг и вершин, а на рис. 11.6 (б) показан граф G_0, построенный в соответствии с приведенным описанием. Так как полный поток, входящий в вершину x_j^+, необходимо должен протекать по дуге (x_j^+, x_j^-) с пропускной способностью w_j, то максимальный поток в графе G с пропускными способностями дуг и вершин равен максимальному потоку в графе G_0, имеющем только пропускные способности дуг. Следует заметить, что если минимальный разрез



(а)



(б)

Рис. 11.6. (а) Граф, вершинам и дугам которого приписаны пропускные способности. (б) Эквивалентный граф, в котором пропускные способности имеют только дуги.

в G_0 не содержит дуги вида (x_j^+, x_j^-), то пропускные способности вершин в G пассивны и излишни. Если же минимальный разрез в G_0 содержит такие дуги, то соответствующие вершины в G насыщены полученным максимальным потоком.

3.3 Графы, у которых пропускные способности дуг ограничены сверху и снизу

Рассмотрим граф $G = (X, A)$, дуги (x_i, x_j) которого имеют пропускные способности (верхние границы потока), скажем, q_{ij}, а также имеют нижние границы потока, скажем, r_{ij}. Допустим,

что мы хотим узнать, существует ли допустимый поток между s и t, т. е. поток ξ_{ij}, удовлетворяющий для всех дуг (x_i, x_j) из G условию (11.1) и, кроме того, неравенствам $r_{ij} \leqslant \xi_{ij} \leqslant q_{ij}$.

Начнем с введения в графе G нового искусственного источника s_a и искусственного стока t_a и построим новый граф G_a. Для каждой дуги (x_i, x_j), у которой $r_{ij} \neq 0$, введем дуги (s_a, x_j) и (x_i, t_a) с пропускными способностями r_{ij} и с нижней границей 0. Уменьшим q_{ij} до $q_{ij} - r_{ij}$, а r_{ij} до 0. Кроме того, введем дугу (t, s) с $q_{ts} = \infty$ и $r_{ts} = 0$.

Применяя, например, эти преобразования к графу, изображенному на рис. 11.7 (а) и имеющему только две дуги с ненулевыми нижними границами, получим граф, показанный на рис. 11.7 (б).

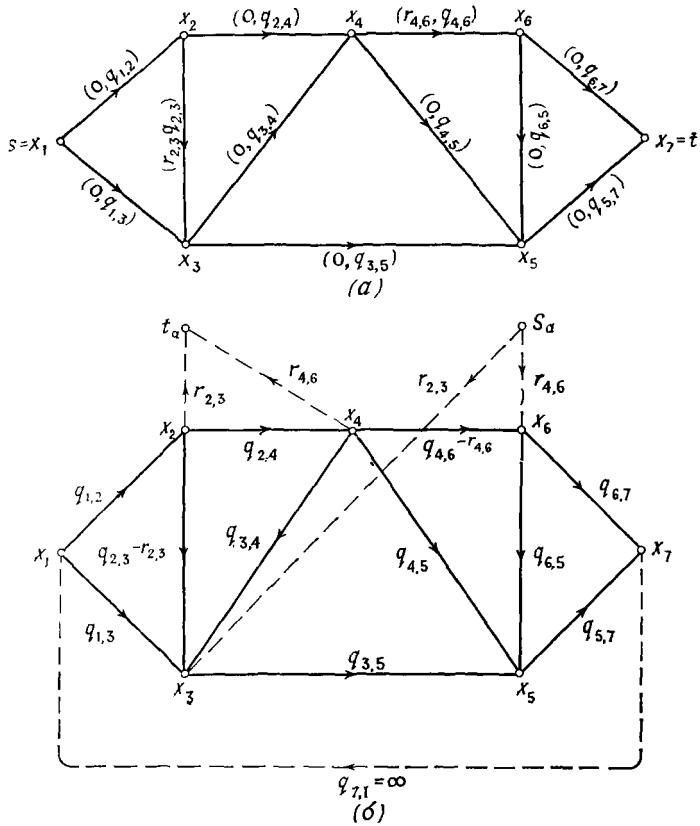


Рис. 11.7. (а) Граф с верхними и нижними границами потоков, проходящих по дугам. (б) Преобразованный граф, в котором для потоков имеются лишь верхние границы (даны значения верхних границ, все нижние границы равны нулю).

Найдем теперь максимальный поток между искусственными вершинами s_a и t_a преобразованного графа G_a. Если значение максимального потока (от s_a к t_a) есть $\sum_{r_{ij}\neq 0} r_{ij}$ (т. е. если все дуги, выходящие из s_a, и все дуги, входящие в t_a, насыщены) и, скажем, поток по дуге (t, s) равен ξ_{ts}, то в первоначальном графе существует допустимый поток со значением ξ_{ts}. В этом можно убедиться так. Если мы вычитаем r_{ij} из потока в дугах (x_i, t_a) и (s_a, x_j) и добавляем r_{ij} к потоку в дуге (x_i, x_j), то значение полного потока от s_a к t_a уменьшается на величину r_{ij}, потоки по дугам (x_i, t_a) и (s_a, x_j) уменьшаются до нуля, а поток по дуге (x_i, x_j) принимает значение ξ_{ij}, $r_{ij} \leqslant \xi_{ij} \leqslant q_{ij}$ (конечное значение ξ_{ij} равно r_{ij}, если первоначальное значение ξ_{ij}, соответствующее максимальному потоку, равно нулю и конечное значение ξ_{ij} равно q_{ij}, если первоначальное значение ξ_{ij} является максимально возможным, т. е. $q_{ij} - r_{ij}$). Шаг, состоящий в вычитании потока из максимального потока, противоположен шагу увеличения в алгоритме максимального потока. Так как мы допустили, что максимальный поток от s_a к t_a равен $\sum_{r_{ij}\neq 0} r_{ij}$, то процесс вычитания потока приведет в конечном итоге к нулевому потоку от s_a к t_a (сделав тем самым эти две искусственные вершины и инцидентные им дуги излишними) и поток во всех дугах с $r_{ij} \neq 0$ будет лежать в пределах $r_{ij} \leqslant \xi_{ij} \leqslant q_{ij}$. Конечный результат является, следовательно, «циркуляцией» потока в графе, причем значение этого потока равно ξ_{ts}. С другой стороны, справедлива

Теорема 3. *Если значение максимального потока (от s_a к t_a) в графе G_a не равно $\sum_{r_{ij}\neq 0} r_{ij}$, то в графе G не существует никакого допустимого потока.*

Доказательство этой теоремы предлагаем в качестве упражнения читателю (см. задачу 5).

4. Максимальный поток между каждой парой вершин

В некоторых случаях, например когда граф представляет телефонную сеть, вершины которой соответствуют станциям, а дуги — телефонным линиям, может потребоваться найти максимальную скорость передачи сообщений между двумя *любыми* станциями s и t. При этом предполагается, что в любой фиксированный момент времени каждая станция может быть связана с произвольной другой, но лишь с одной станцией. В данном примере пропускная способность кабеля равна числу независимых разговоров,

которые могут вестись по нему. Подобная ситуация имеет место и в случае, когда граф представляет сеть дорог с односторонним движением и требуется определить, как одностороннее движение влияет на максимум транспортного потока между двумя районами, если эти районы рассматриваются обособленно друг от друга.

Как было отмечено во введении, эта задача может быть решена просто с помощью задачи о максимальном потоке (от s к t) для всех пар вершин (s, t). Однако такой метод излишне трудоемок, поэтому мы изложим здесь алгоритм Гомори и Ху [14], который в случае *неориентированных* графов намного более эффективен. Так как этот алгоритм использует два важных понятия, то мы сначала обсудим их, чтобы сделать более ясным описание самого алгоритма.

4.1. Потоковая эквивалентность

Теорема 4. *Пусть f_{ij} — значение максимального потока от вершины x_i к вершине x_j графа G. (Так как G — неориентированный граф, то $f_{ij} = f_{ji}$). Эти потоки удовлетворяют соотношению*

$$f_{ij} \geqslant \min [f_{ik}, f_{kj}] \quad \textit{для всех } i, j, k. \tag{11.8}$$

Доказательство. Если $(X_0, \bar{X}_0)$ — минимальный разрез, дающий f_{ij}, т. е.

$$f_{ij} = v(X_0, \bar{X}_0), \qquad x_i \in X_0, \quad x_j \in \bar{X}_0,$$

то при $x_k \in X_0$ выполняется неравенство $f_{ij} \geqslant f_{kj}$, так как $(X_0, \bar{X}_0)$ является также разрезом, отделяющим x_k от x_j, а при $x_k \in \bar{X}_0$ справедливо неравенство $f_{ij} \geqslant f_{ik}$, так как $(X_0, \bar{X}_0)$ является также разрезом, отделяющим x_i от x_k. Отсюда немедленно вытекает теорема.

Повторно применяя неравенство (11.8) к потокам в скобках из (11.8), получим

$$f_{ij} \geqslant \min [f_{ik_1}, f_{k_1k_2}, f_{k_2k_3}, \ldots, f_{k_pj}] \tag{11.9}$$

для любой последовательности вершин $x_i, x_{k_1}, x_{k_2}, x_{k_3}, \ldots, x_{k_p}, x_j$.

Если теперь рассмотреть гипотетический полный граф G' с n вершинами и взять в качестве «длины» ребра (x'_k, x'_j) максимальный поток f_{ij} между соответствующими вершинами x_i и x_j из G, то эти «длины» можно использовать для построения «длиннейшего остова» T^* графа G', применяя один из методов построения кратчайшего (или длиннейшего) остова, приведенных в гл. 7. В соответствии с (11.9) можно сказать, что поток f_{ij} больше или равен, чем «длина» кратчайшего ребра, лежащего на единственной цепи в T^* из x'_i в x'_j. Если, однако, f_{ij} больше, чем «длина» кратчайшего ребра, то удаление этого ребра и добавление (x'_i, x'_j) даст новое

дерево из G', более длинное, чем T^*, что противоречит предположению о максимальности T^*. Таким образом, f_{ij} должно быть *равно* «длине» кратчайшего ребра на единственной цепи из x_i' в x_j' в дереве T^*. Следовательно, древовидный граф T^*, пропускные способности дуг которого равны «длинам» ребер из G', будет *потоково эквивалентен* первоначальному графу G. Под потоковой эквивалентностью понимается следующее свойство: если графы G и T рассматриваются как «черные ящики» с вершинами «выведенными наружу», то эти два графа будут неразличимы при характеризации их с помощью максимальных потоков между парами вершин. Вышеприведенная конструкция графа T^* является предельно общей и показывает, что для любого графа G всегда существует потоково эквивалентный древовидный граф T^*.

4.2. Конденсация вершин

Предположим, что для графа $G = (X, \Gamma)$ решена задача о максимальном потоке (от s к t), причем s и t — две случайным образом выбранные вершины. Пусть $(X_0, \bar{X}_0)$ — минимальный разрез, соответствующий максимальному потоку, и рассмотрим две вершины x_i и x_j, лежащие обе в X_0 (или обе лежащие в $\bar{X}_0$). Если мы теперь хотим найти f_{ij}, максимальный поток из x_i в x_j, то все вершины из $\bar{X}_0$ (или X_0, если x_i и $x_j \in \bar{X}_0$) могут быть «сконденсированы» в одну вершину $\tilde{x}_0$ коль скоро речь идет только о вычислении потока. Эта конденсация такова, что ребра (x_a, x_b), x_a и $x_b \in X_0$ заменяются ребрами $(x_a, \tilde{x}_0)$ и любые параллельные ребра между одной и той же парой вершин (которые могут получиться) заменяются единственным ребром, пропускная способность которого равна сумме пропускных способностей параллельных ребер. На рис. 11.8 (а), (б), (в) иллюстрируется процесс конденсации. Тот факт, что такая конденсация множества $\bar{X}_0$ возможна, был установлен Гомори и Ху [14, 16], развившими вышеописанный нами метод. В доказательстве показывается, что все вершины $\bar{X}_0$ должны лежать с одной и той же стороны от минимального разреза $(Y_0, \bar{Y}_0)$, отделяющего x_i от x_j (т. е. или $\bar{X}_0 \subseteq Y_0$, или $\bar{X}_0 \subseteq \bar{Y}_0$), так что внутренние свойства подграфа (X_0, Γ) графа G не влияют на вычисление минимального разреза между x_i и x_j.

4.3. Алгоритм построения максимального потока между всеми парами вершин

Алгоритм, описанный в настоящем разделе, порождает дерево T^*, которое потоково эквивалентно неориентированному графу G. Максимальный поток f_{ij} между двумя вершинами x_i и x_j

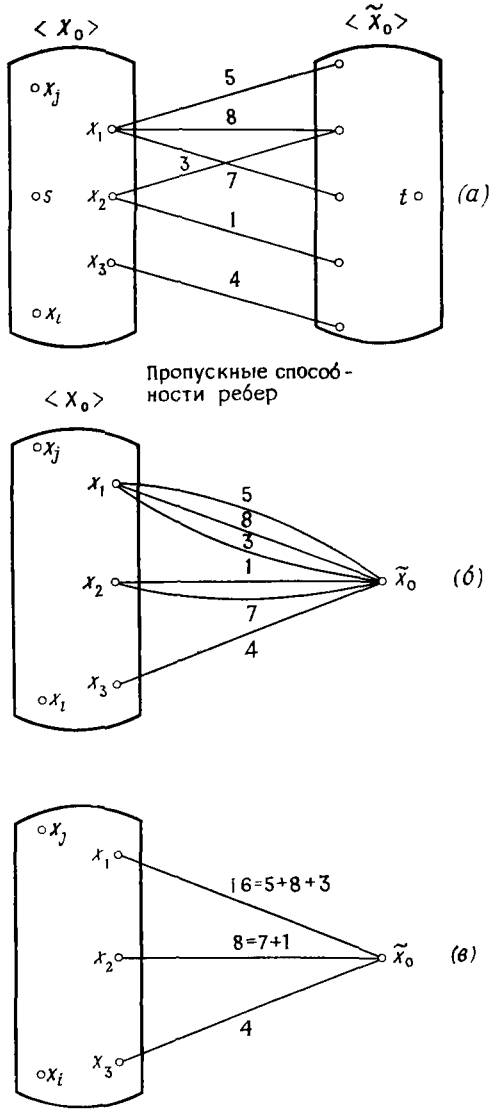


Рис. 11.8. (а) **Минимальный** (от s к t) разрез. (б) Конденсация вершин при вычислении потока от x_i к x_j. (в) Сконденсированный граф.

графа G может быть следующим образом найден по этому дереву:

$$f_{ij} = \min [q'_{ik_1}, q'_{k_1k_2}, q'_{k_2k_3}, \ldots, q'_{k_pj}], \qquad (11.10)$$

где $(x'_i, x'_{k_1}, x'_{k_2}, \ldots, x'_j)$ — единственная цепь, проходящая по ребрам дерева T^* и ведущая от x'_i к x'_j. Каждая вершина x'_k из T^* соответствует вершине x_k из G, а q'_{kl} является пропускной способностью ребра (x'_k, x'_l) из T^*.

Идея алгоритма проста: так как существует дерево T^*, потоково эквивалентное графу G, и так как T^* содержит только $n-1$ ребер, то все, что́ необходимо,— это вычислить пропускные способности $n-1$ ребер дерева T^*. Описанный в данном разделе алгоритм вычисляет пропускные способности ребер дерева T^* за $n-1$ этапов. Каждый этап состоит в вычислении потока в графе, получаемом последовательно из ранее построенного графа с помощью конденсации, как об этом говорилось выше в разд. 4.2.

4.3.1. Описание алгоритма. Так как алгоритм порождает дерево T^* постепенно и так как на любом из $(n-1)$ этапов действия алгоритма «вершины» из T^* могут быть на самом деле множествами вершин из G, то мы будем, во избежание недоразумений, называть вершины из G G-вершинами, а вершины из T^* — T^*-вершинами. Ниже дается достаточно комментариев, чтобы сделать алгоритм понятным.

Шаг 1. Положить $S_1 = X$, $N = 1$.

На любом этапе T^* является графом, определенным N T^*-вершинами $S_1, S_2, \ldots, S_N$, и каждая из них соответствует некоторому множеству G-вершин. Вначале граф T^* состоит из единственной вершины S_1.

Шаг 2. Найти множество $S^* \in \{S_1, S_2, \ldots, S_N\}$, содержащее более чем одну G-вершину. Если такого не существует, то перейти к шагу 6, в противном случае — к шагу 3.

Шаг 3. Если бы S^* было удалено из T^*, то дерево распалось бы на некоторое число поддеревьев (связанных компонент). Сконденсировать T^*-вершины в каждом поддереве в одну вершину и образовать сконденсированный граф. Взять любые две вершины x_i и $x_j \in S^*$ и найти минимальный разрез $(X_o, \bar{X}_o)$ в G, отделяющий x_i от x_j, с помощью вычисления максимального потока (от x_i к x_j).

Шаг 4. Удалить из T^* T^*-вершину S^* вместе с ребрами, ей инцидентными, и заменить ее на две T^*-вершины, составленные из G-вершин множеств $S^* \cap X_o$ и $S^* \cap \bar{X}_o$, и на ребро между ними с пропускной способностью $v(X_o, \bar{X}_o)$. Итак, для всех T^*-вершин S_i, которые до замены были инцидентны S^* (скажем, с пропускной способностью ребра c_i^o), добавить к T^* ребро $(S_i, S^* \cap X_o)$, если $S_i \subset X_o$, или же к T^* добавить ребро

$(S_i,\ S^* \cap \widetilde{X}_o)$, если $S_i \subset \widetilde{X}_o$. Пропускные способности ребер и в том и в другом случае взять равными c_i^o.

Замечание. Как уже было отмечено в разд. 4.2, Гомори и Ху [14] показали, что S_i лежит целиком или в X_o, или в $\widetilde{X}_o$, так что возможны только два указанных выше случая.

Шаг 5. Положить $N = N + 1$. Вершины из T^* являются теперь множествами G-вершин $S_1, S_2, \ldots, S^* \cap X_o, S^* \cap \cap \widetilde{X}_o, \ldots, S_N$, где S^* заменена двумя T^*-вершинами $S^* \cap X_o$ и $S^* \cap \widetilde{X}_o$, как объяснялось выше. Перейти к шагу 2.

Шаг 6. Останов. T^* будет теперь требуемым потоково эквивалентным графу G деревом, и его T^*-вершины являются теперь единственными G-вершинами. Пропускные способности ребер из T^* соответствуют значениям $(n-1)$ независимых разрезов в графе G. Равенство (11.10) можно теперь использовать для вычисления f_{ij} (для любых $x_i, x_j \in X$) непосредственно из T^*.

То обстоятельство, что вышеприведенный алгоритм дает оптимальный результат, следует непосредственно из свойств минимальных разрезов и свойств потоково эквивалентного графу G дерева, приведенных выше в разд. 4.1 и 4.2. Формальное доказательство можно найти в книге Ху [16].

4.4. Пример

Рассмотрим неориентированный граф G, изображенный на рис. 11.9. Пропускные способности показаны цифрами, стоящими

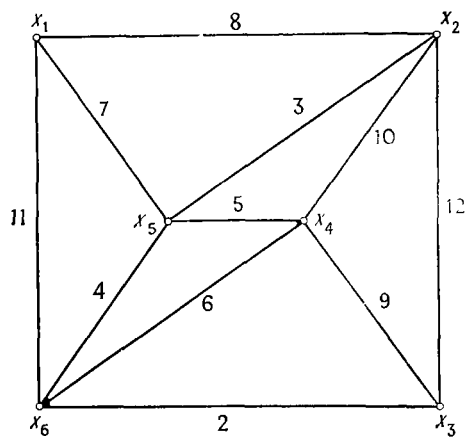


Рис. 11.9. Граф из примера 4.4.

у ребер. Нужно найти максимальный поток между каждой парой вершин графа G. Применим вышеописанный алгоритм.

Шаг 1. $S_1 = \{x_1, x_2, x_3, x_4, x_5, x_6\}$; $N = 1$.

Шаг 2. $S^* = S_1$.

Шаг 3. Граф не может быть сконденсирован. Берем (произвольно) $x_i = x_1$ и $x_j = x_2$. Вычисляя максимальный поток (от x_1 к x_2), найдем, что минимальным разрезом будет $(X_0, \bar{X}_0)$, где $X_0 = \{x_1, x_5, x_6\}$ и $\bar{X}_0 = \{x_2, x_3, x_4\}$, а значение разреза равно 24.

Шаг 4. Дерево T^* и пропускные способности его ребер изображены на рис. 11.10а.

Шаг 5. $N = 2$ (т. е. T^* имеет теперь, как показано на рис. 11.10а, две вершины: S_1 и S_2).

Шаг 2. Берем $S^* = S_2$.

Шаг 3. Выбираем (произвольно) $x_i = x_3$ и $x_j = x_4$. Полученный затем сконденсированный граф изображен на рис. 11.10б. Вычисляя для него максимальный поток, порождаем минимальный разрез $(X_0, \bar{X}_0)$, где $X_0 = \{x_3\}$ и $\bar{X}_0 = \{\underbrace{x_1, x_5, x_6}_{s_1}, x_2, x_4\}$. Значение этого разреза равно 23.

Шаг 4. T^*-вершина $S_2 = \{x_2, x_3, x_4\}$ заменяется теперь двумя новыми T^*-вершинами $\{x_3\}$ и $\{x_2, x_4\}$ и ребром между ними со значением 23. Так как $S_1 \subset \bar{X}_0$, то ребро (S_1, S_2) удаляется и заменяется ребром (S_1, S_3), где $S_3 = \{x_2, x_4\}$.

Шаг 5. $N = 3$ (т. е. T^* имеет теперь 3 вершины, как показано на рис. 11.10в).

Шаг 2. Возьмем $S^* = S_3$.

Шаг 3. Берем $x_i = x_2$ и $x_j = x_3$. Сконденсированный граф изображен на рис. 11.10г. Минимальный разрез со значением 30 равен $(X_0, \bar{X}_0)$, где $X_0 = \{x_4\}$ и $\bar{X}_0 = \{x_2, x_3, x_1, x_5, x_6\}$.

Шаг 4. T^*-вершина $S_3 = \{x_2, x_4\}$ теперь заменяется двумя новыми T^*-вершинами $\{s_2\}$ и $\{x_4\}$, а новое дерево показано на рис. 11.10д.

Продолжая подобным образом и беря $S^* = S_1$, последовательно получаем потоково эквивалентные деревья, изображенные на рисунках 11.10е, ж, з, и. Если использовать рис. 11.10(и), то матрицу максимальных потоков $[f_{ij}]$ первоначального графа можно

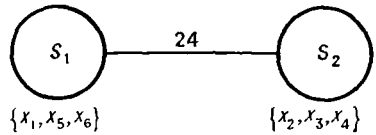


Рис. 11.10а. T^* после 1-го этапа.

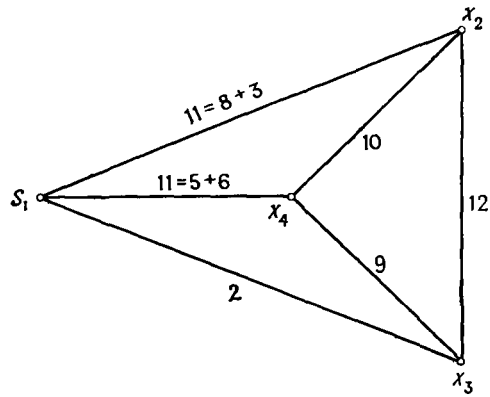


Рис. 11.10б и г. Сконденсированный граф после 2-го (б) и 3-го (г) этапов.

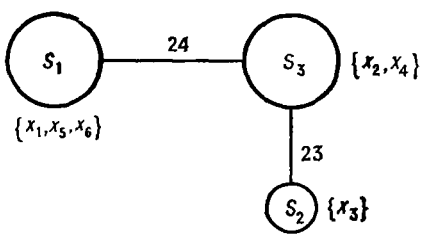


Рис. 11.10в. T^* после 2-го этапа.

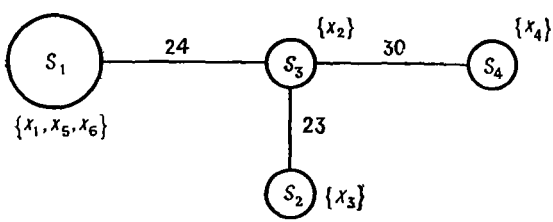


Рис. 11.10д. T^* после 3-го этапа.

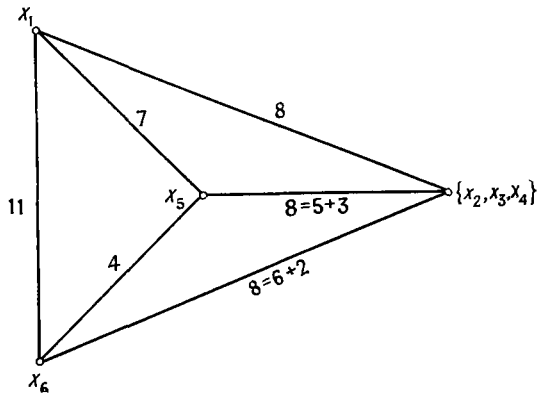


Рис. 11.10е и з. Сконденсированный граф после 4-го (е) и 5 го (з) этапов.

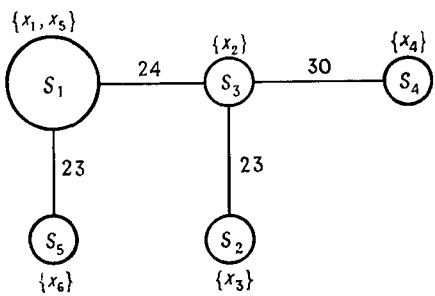


Рис. 11.10ж. T^* после 4-го этапа.

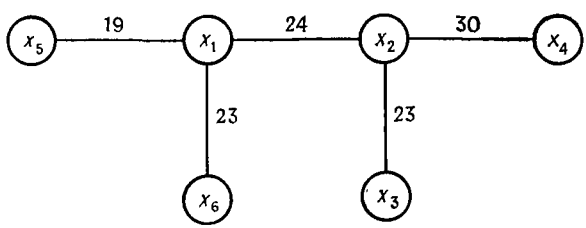


Рис. 11.10и. Окончательное потоково эквивалентное дерево T^*.

вычислить непосредственно с помощью (11.10). Она имеет вид

Матрица максимальных потоков

	x_1	x_2	x_3	x_4	x_5	x_6
x_1	—	24	23	24	19	23
x_2	24	—	23	30	19	23
x_3	23	23	—	23	19	23
x_4	24	30	23	—	19	23
x_5	19	19	19	19	—	19
x_6	23	23	23	23	19	—

Все 5 (в общем случае $n - 1$) разрезов графа G, соответствующие ребрам дерева T^*, показаны пунктиром на рис. 11.11.

Следует заметить, что, вообще говоря, потоково эквивалентное дерево T^* не является единственным и что существуют другие

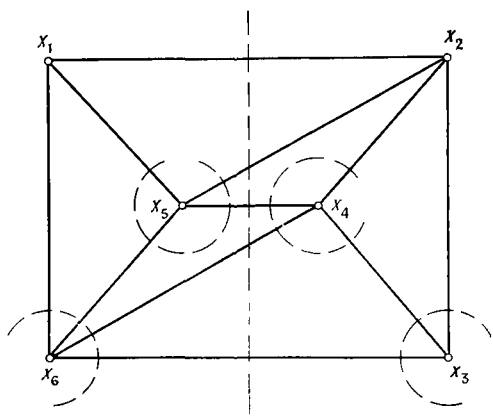


Рис. 11.11. Разрез, дающий дерево T^*.

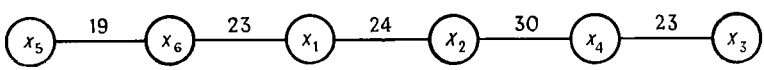


Рис. 11.12. Другое потоково эквивалентное дерево.

деревья, также потоково эквивалентные графу G. Для нашего примера одно из таких деревьев, которое на самом деле будет гамильтоновой цепью, показано на рис. 11.12.

5. Поток минимальной стоимости от s к t

В разделе 2 мы рассмотрели задачу максимизации потока от s к t безотносительно к какой-либо стоимости. Теперь мы рассмотрим задачу нахождения такого потока заданной величины v от s к t, стоимость которого минимальна. В этой задаче каждая дуга (x_i, x_j) имеет два связанных с ней числа — пропускную способность q_{ij} и стоимость c_{ij} единицы потока по этой дуге.

Очевидно, что если v больше, чем значение максимального потока от s к t, то не существует никакого решения. Если же v не превышает этого значения, то, вообще говоря, может существовать несколько различных потоков и требуется найти поток величины v, обеспечивающий минимальную стоимость. Наиболее известной процедурой решения задачи о потоке с минимальной стоимостью является так называемый алгоритм «беспорядка» Форда и Фалкерсона, и интересующийся читатель может найти описание этого эффективного алгоритма в [12], [4] и [7]. Алгоритмы, которые мы здесь опишем, принадлежат Клейну [20] и Басакеру и Гауэну [6]. Эти алгоритмы концептуально проще, чем метод беспорядка, и используют уже введенную технику. С вычислительной точки зрения эти методы сравнимы [2].

5.1. Алгоритм, основанный на выявлении отрицательных циклов

Предположим, что в графе существует допустимый поток ξ со значением v и что этот поток известен. Такой поток может быть получен с помощью алгоритма максимального потока (от s к t) из разд. 2 и добавления потока общей величины $\delta(t)$ на всех аугментальных цепях (шаги с 4 по 6 алгоритма) до тех пор, пока поток f_{st} не достигнет значения v, которое по условию меньше значения максимального потока.

Для этого допустимого потока определим новый инкрементальный граф $G^{\mu}(\xi) = (X^{\mu}, A^{\mu})$ в точности так, как это было объяснено в разд. 2.3, со следующими стоимостями дуг.

Для каждой дуги $(x_i^{\mu}, x_j^{\mu}) \in A_1^{\mu}$ положим $c_{ij}^{\mu} = c_{ij}$.

Для каждой дуги $(x_j^{\mu}, x_i^{\mu}) \in A_2^{\mu}$ положим $c_{ji}^{\mu} = -c_{ij}$.

Новый граф $G^{\mu}(\xi)$ дает теперь инкрементальные пропускные способности и стоимости (относительно начального потока ξ) любого дополнительного потока, введенного в G. Алгоритм основан на следующей теореме.

Теорема 5. *Поток ξ будет потоком минимальной стоимости со значением v тогда и только тогда, когда в $G^{\mu}(\xi)$ не существует никакого цикла Φ, сумма стоимости дуг которого отрицательна.*

Доказательство. Пусть $c[\xi]$ — стоимость потока ξ в графе G, а $c[\Phi \mid G^{\mu}(\xi)]$ — сумма стоимости дуг в цикле Φ по отношению к графу $G^{\mu}(\xi)$.

Необходимость. Пусть $c[\Phi \mid G^{\mu}(\xi)] < 0$ для некоторого цикла Φ из $G^{\mu}(\xi)$. Циркуляция дополнительной единицы потока по циклу Φ дает новый поток $\xi + 1 \circ (\Phi)$, оставляя значение v потока от s к t неизменным. Стоимость потока $\xi + 1 \circ (\Phi)$ равна $c[\xi] + c[\Phi] G^{\mu}(\xi)] < c[\xi]$, а это противоречит допущению, что ξ — поток минимальной стоимости со значением v.

Достаточность. Допустим, что в графе $G^{\mu}(\xi)$: $[\Phi \mid G(\xi)] \geqslant 0$ для каждого цикла Φ и что ξ^* ($\neq \xi$) — поток минимальной стоимости со значением v. Обозначим теперь через $\xi^* - \xi$ поток, значение которого на дуге (x_i, x_j) равно $\xi^*_{ij} - \xi_{ij}$.

Так как ξ^* и ξ можно разложить в сумму потоков (от s к t) по цепям в G, то построение унитарного графа G^e (см. разд. 2.4) для потока $\xi^* - \xi$ дает следующие полустепени вершин:

$$d_0^{G^e}(x_i) = d_t^{G^e}(x_i), \qquad \forall x_i \in X.$$

Так как в соответствии с разд. 2.4 G^e состоит из наборов конформальных единичных потоков по циклам $\Phi_1, \Phi_2, \ldots, \Phi_k$ (например), то мы можем написать

$$\xi^* - \xi = 1 \circ (\Phi_1) + 1 \circ (\Phi_2) + \ldots + 1 \circ (\Phi_k).$$

Так как потоки $1 \circ (\Phi_i)$, $i = 1, \ldots, k$, попарно конформальны и нам известно, что поток $\xi^* = \xi + 1 \circ (\Phi_1) + \ldots + 1 \circ (\Phi_k)$, допустим, то любая сумма $\xi + 1 \circ (\Phi_1) + \ldots + 1 \circ (\Phi_l)$ будет допустимой для любых l, $1 \leqslant l \leqslant k$. Таким образом, рассматривая поток $\xi + 1 \circ (\Phi_1)$, получаем

$$c[\xi + 1 \circ (\Phi_1)] = c[\xi] + c[\Phi_1 \mid G^{\mu}(\xi)] \geqslant c[\xi].$$

Рассмотрим теперь инкрементальный граф $G^{\mu}(\xi + 1 \circ (\Phi_1))$. Единственными дугами этого графа, стоимости которых уменьшились по сравнению с соответствующими стоимостями в графе $G^{\mu}(\xi)$, являются те, которые будут «обращениями» дуг в Φ_1. Но, так как потоки $1 \circ (\Phi_1)$, $1 \circ (\Phi_2)$, ... конформальны, такие дуги не могут быть использованы ни одним из оставшихся циклов $\Phi_2, \ldots, \Phi_k$ и, следовательно, не влияют на последующие рассуждения.

Теперь мы имеем

$$c[\Phi_l \mid G^{\mu}(\xi + 1 \circ (\Phi_1))] \geqslant c[\Phi_l \mid G^{\mu}(\xi)]$$

для любого $l = 2, \ldots, k$.

Стоимость потока $\xi + 1 \circ (\Phi_1) + 1 \circ (\Phi_2)$ равна тогда

$$c[\xi + 1 \circ (\Phi_1) + 1 \circ (\Phi_2)] = c[\xi + 1 \circ (\Phi_1)] + c[\Phi_2 | G^{\mu}(\xi + 1 \circ (\Phi_1))] \geqslant$$
$$\geqslant c[\xi + 1 \circ (\Phi_1)] + c[\Phi_2 | G^{\mu}(\xi)] \geqslant c[\xi + 1 \circ (\Phi_1)] \geqslant c[\xi].$$

Продолжая рассуждения аналогичным образом, мы получим окончательно $c[\xi^*] \geqslant c[\xi]$, а это противоречит предположению о том, что ξ^* — поток минимальной стоимости со значением v. Теорема доказана.

Таким образом, в соответствии с теоремой 5, все, что требуется для нахождения потока минимальной стоимости со значением v,— это начать с допустимого потока ξ со значением v, построить граф $G^{\mu}(\xi)$ и проверить, нет ли в нем циклов с отрицательной стоимостью, используя любой из алгоритмов нахождения кратчайшей цепи, данных в разд. 3 и 4 гл. 9. Если не существует никакого цикла с отрицательной стоимостью, то поток будет потоком минимальной стоимости. Если цикл Ф с отрицательной стоимостью существует, то надо «пустить» по нему поток с максимально возможным значением δ. Суммарный поток от s к t не изменит своего значения v, а его стоимость уменьшится на $\delta \cdot c(\Phi)$, где $c(\Phi)$ — величина стоимости отрицательного цикла. Очевидно, δ должно быть выбрано так, чтобы пропускные способности дуг в $G^{\mu}(\xi)$ не превышались, т. е.

$$\delta = \min_{(x_i^{\mu}, x_j^{\mu}) \text{ в } \Phi} [q_{ij}^{\mu}]. \tag{11.11}$$

Если этот поток δ наложить на поток ξ уже в графе G, то, в силу первоначального выбора пропускных способностей дуг в $G^{\mu}(\xi)$, результирующий поток все еще будет допустимым. Процесс можно повторять, начиная его с полученного нового потока ξ, строя новый граф $G^{\mu}(\xi)$ относительно нового потока и выявляя в этом графе циклы с отрицательной стоимостью. Описание алгоритма дано ниже.

5.1.1. Описание алгоритма

Шаг 1. Использовать алгоритм максимального потока (от s к t) из разд. 2 для нахождения в графе G допустимого потока ξ со значением v.

Шаг 2. Построить для этого потока граф $G^{\mu}(\xi)$, пользуясь данными ранее правилами.

Шаг 3. Отправляясь от матрицы стоимостей графа $G^{\mu}(\xi)$ и используя алгоритм кратчайшей цепи (см. разд. 3.1 и 4 гл. 8), определить, существует ли в графе $G^{\mu}(\xi)$ цикл с отрицательной стоимостью. Если такой цикл существует, то найти его (цикл Ф)

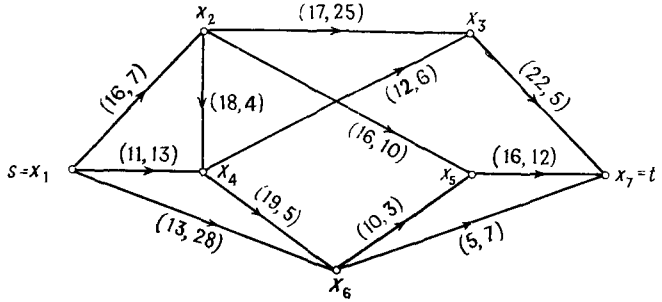


Рис. 11.13. Граф из примера 5.1.2. Первая пометка является пропускной способностью дуги, вторая — стоимостью дуги.

и перейти к шагу 5. Если такого цикла нет, то остановиться. Поток ξ будет потоком минимальной стоимости.

Шаг 4. По (11.11) вычислить δ.

(I) Для всех (x_i^μ, x_j^μ) в Φ с $c_{ij}^\mu < 0$ изменить поток ξ_{ji} в соответствующей дуге (x_j, x_i) из G с ξ_{ji} на $\xi_{ji} - \delta$.

(II) Для всех (x_i^μ, x_j^μ) в Φ с $c_{ij}^\mu > 0$ изменить поток ξ_{ij} в соответствующей дуге (x_i, x_j) из G с ξ_{ij} на $\xi_{ij} + \delta$.

Шаг 5. Повторить шаг 2, начиная с нового потока ξ.

5.1.2. **Пример.** Рассмотрим граф G, изображенный на рис. 11.13, где первое число пометки у каждой дуги равно ее пропускной способности, а второе — стоимости. Мы хотим найти поток от s к t минимальной стоимости со значением 20.

Для выявления циклов с отрицательной стоимостью на шаге 3 вышеописанного метода используем алгоритм кратчайшей цепи Флойда (разд. 3.1 гл. 8).

Шаг 1. Алгоритм максимального потока дает первый допустимый поток, изображенный на рис. 11.14а. Стоимость этого потока

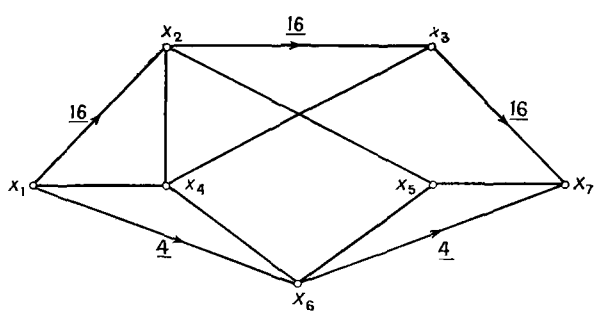


Рис. 11.14а. Начальный поток ξ.

равна

$$16\,(7 + 25 + 5) + 4\,(28 + 7) = 732.$$

Шаг 2. Исходя из этого потока, построим граф $G^{\mu}(\xi)$, как показано на рис. 11.14б, где цикл отрицательной стоимости изображен пунктиром.

Шаг 3. Начиная с матрицы стоимостей

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1	0			13		28	
x_2	−7	0	25	4	10		
x_3		−25	0				5
x_4			6	0		5	
x_5					0		12
x_6	−28				3	0	7
x_7			−5			−7	0

(где на пустых местах должны стоять элементы ∞) и применяя алгоритм Флойда, получаем следующие матрицы наименьших стоимостей вместе с соответствующими им матрицами цепей:

Матрицы в конце первой итерации ($k=1$)

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
1	0			13		28	
2	−7	0	25	4	10	21	
3		−25	0				5
4			6	0		5	
5					0		12
6	−28			−15	3	0	7
7			−5			−7	0

Матрица наименьших стоимостей

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1	1	1	1	1	1	1	1
x_2	2	2	2	2	2	1	2
x_3	3	3	3	3	3	3	3
x_4	4	4	4	4	4	4	4
x_5	5	5	5	5	5	5	5
x_6	6	6	6	1	6	6	6
x_7	7	7	7	7	7	7	7

Матрица цепей

В последней итерации элемент $c^k_{4,4}$ получается отрицательным, со значением -15, которое показывает существование цикла отрицательной стоимости, содержащего вершину x_4. Этот цикл, найденный по матрице цепей, имеет вид (x_4, x_3, x_2, x_4).

Шаг 4. Из (11.11) находим значение δ:

Матрицы в конце второй итерации (k=2)

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1	0			13		28	
x_2	−7	0	25	4	10	21	
x_3	−32	−25	0	−21	−15	−4	5
x_4			6	0		5	
x_5					0		12
x_6	−28			−15	3	0	7
x_7			−5			−7	0

Матрица наименьших стоимостей

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1	1	1	1	1	1	1	1
x_2	2	2	2	2	2	1	2
x_3	2	3	3	2	2	1	3
x_4	4	4	4	4	4	4	4
x_5	5	5	5	5	5	5	5
x_6	6	6	6	1	6	6	6
x_7	7	7	7	7	7	7	7

Матрица цепей

Матрицы в конце третьей итерации (k=3)

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1	0			13		28	
x_2	−7	0	25	4	10	21	30
x_3	−32	−25	0	−21	−15	−4	5
x_4	−26	−19	6	−15	−9	2	11
x_5					0		12
x_6	−28			−15	3	0	7
x_7	−37	−30	−5	−26	−20	−9	0

Матрица наименьших стоимостей

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1	1	1	1	1	1	1	1
x_2	2	2	2	2	2	1	3
x_3	2	3	3	2	2	1	3
x_4	2	3	4	2	2	1	3
x_5	5	5	5	5	5	5	5
x_6	6	6	6	6	6	6	6
x_7	2	3	7	2	2	1	7

Матрица цепей

Новый поток, полученный после введения в цикл циркуляционного потока δ, изображен на рис. 11.14в. Стоимость нового потока равна 552. Возвращаясь к шагу 2, находим новый граф $G^{\mu}(\xi)$, показанный на рис. 11.14г. Поступая как и раньше, получаем

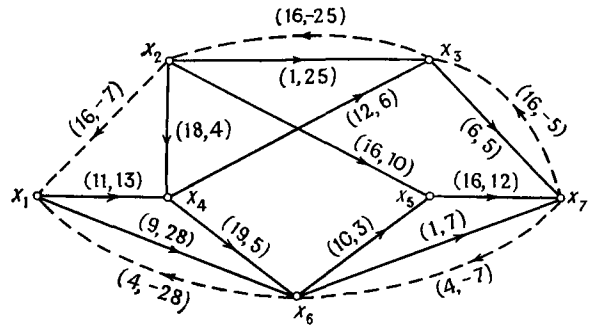


Рис. 11.14б. $G^{\mu}(\xi)$ для потока на рис. (а)·

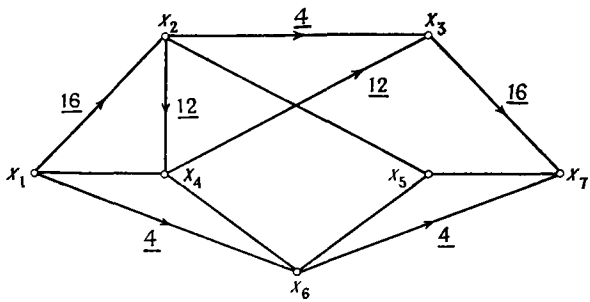


Рис. 11.14в. Улучшенный поток ξ.

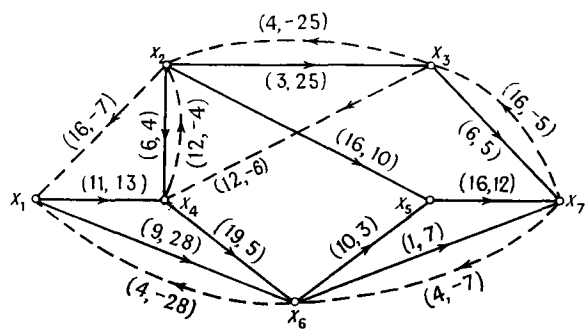


Рис. 11.14г. $G^{\mu}(\xi)$ для потока на рис. (в).

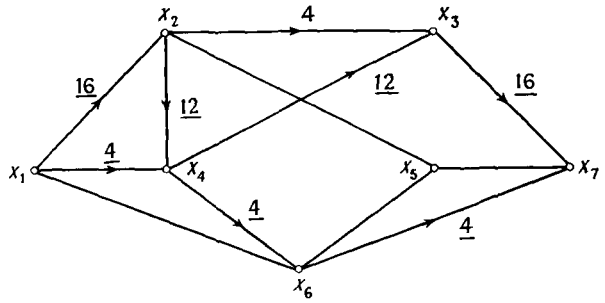


Рис. 11.14д. Улучшенный поток ξ.

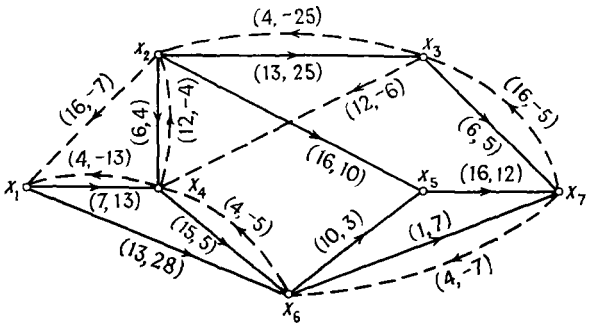


Рис. 11.14е. $G^{\mu}(\xi)$ для потока на рис. (д).

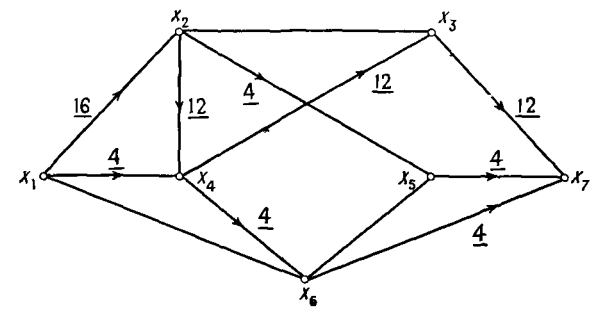


Рис. 11.14ж. Улучшенный поток ξ.

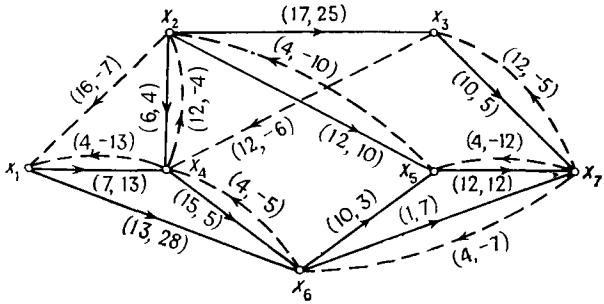


Рис. 11.14з. $G^{\mu}(\xi)$ для потока на рис. (ж).

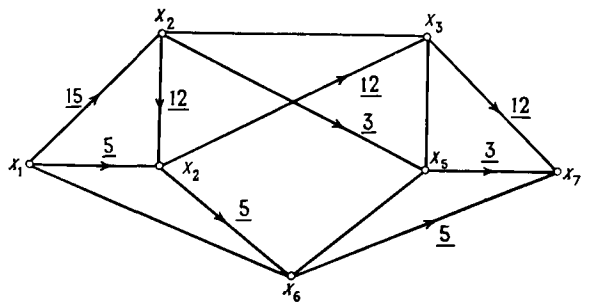


Рис. 11.14и. Улучшенный поток ξ.

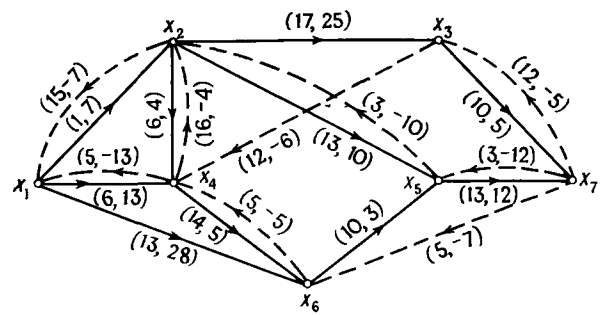


Рис. 11.14к. $G^{\mu}(\xi)$ для потока на рис. (и).

Шаг 3. Обнаружен отрицательный цикл (x_6, x_1, x_4, x_6) со стоимостью -10.

Шаг 4. $\delta = \min [4, 11, 19] = 4$.
Новый поток со стоимостью 512 изображен на рис. 11.14д.

Шаг 2. Граф $G^{\mu}(\xi)$, соответствующий новому потоку, изображен на рис. 11.14е.

Шаг 3. Обнаружен отрицательный цикл $(x_7, x_3, x_2, x_5, x_7)$ со стоимостью -8.

Шаг 4. $\delta = \min [16, 4, 16, 16] = 4$.
Новый поток со стоимостью 480 изображен на рис. 11.14ж.

Шаг 2. Получившийся граф $G^{\mu}(\xi)$ показан на рис. 11.14з.

Шаг 3. Обнаружен отрицательный цикл $(x_1, x_4, x_6, x_7, x_5, x_2, x_1)$ со стоимостью -4.

Шаг 4. $\delta = \min (7, 15, 1, 4, 4, 16) = 1$.
Новый поток со стоимостью 476 изображен на рис. 11.14 и.

Шаг 2. Граф $G^{\mu}(\xi)$ изображен на рис. 11.14к.

Шаг 3. Обнаружен отрицательный цикл (x_1, x_2, x_4, x_1) со стоимостью -2.

Шаг 4. $\delta = \min [1, 6, 5] = 1$.

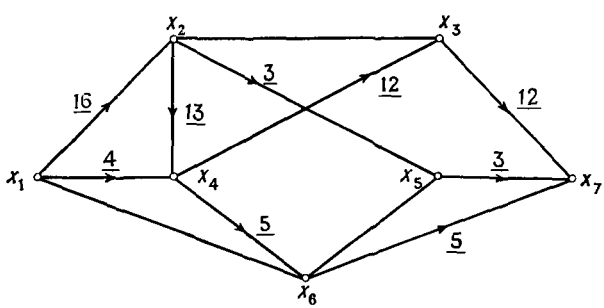


Рис. 11.14л. Поток минимальной стоимости.

Новый поток со стоимостью 474 изображен на рис. 11.14л.
В последнем инкрементальном графе нет никаких циклов с отрицательной стоимостью, и поэтому поток, изображенный на рис. 11.14 *л*, является потоком наименьшей стоимости со значением 20 и его стоимость равна 474.

5.1.3. Шаг 3 из алгоритма 5.1.1. В вышеприведенном примере был использован метод Флойда для нахождения циклов отрицательной стоимости в инкрементальном графе $G^{\mu}(\xi)$ на шаге 3 алгоритма определения потока минимальной стоимости, данного в разд. 5.1.1. Но, как уже отмечалось в разд. 4 гл. 8, это не лучший метод выявления циклов отрицательной стоимости для графа G, в котором существует вершина x_0, достижимая из всех других вершин G. Лучший способ состоит в использовании алгоритма, который находит кратчайшую цепь от x_0 ко всем другим вершинам, а не в применении алгоритма Флойда, находящего кратчайшие цепи между каждой парой вершин [2]. В этом отношении более успешно, как это объясняется в разд. 4 настоящей главы, может быть использован алгоритм кратчайшей цепи из разд. 2.2.1 главы 8.

Для инкрементального графа $G^{\mu}(\xi)$ очень просто показать, что в нем существует по крайней мере одна вершина, из которой можно достичь любую другую вершину графа $G^{\mu}(\xi)$, и что на самом деле конечная вершина t обладает этим свойством. Это можно показать следующим образом.

Так как для любой дуги (x_i, x_j) из G, для которой существует некоторый ненулевой поток, в инкрементальном графе $G^{\mu}(\xi)$ существует дуга (x_j^{μ}, x_i^{μ}), то все вершины x_i^{μ} из $G^{\mu}(\xi)$, соответствующие вершинам x_i из G, лежащим на цепях потока от s к t, могут быть достигнуты из t с использованием «обратных» цепей, образованных дугами (x_j^{μ}, x_i^{μ}). Таким же способом может быть достигнут источник s. Итак, каждая вершина x_i из G может быть достигнута из s (в противном случае x_i может быть удалена из G, и это не повлияет на поток). Следовательно, так как дуги (x_i, x_j), не несущие никакого потока в G, появляются в $G^{\mu}(\xi)$ в качестве дуг (x_i^{μ}, x_j^{μ}), вершины x_i^{μ} из $G^{\mu}(\xi)$, соответствующие вершинам x_i из G, не лежащим на цепях потока, могут быть достигнуты из t через s. Иными словами, сначала просто достигаем s, как было сказано выше, а затем следуем по пути из s в x_i^{μ}.

Поэтому в инкрементальном графе $G^{\mu}(\xi)$ все вершины могут быть достигнуты из t, и если эта вершина используется в качестве источника при нахождении кратчайшей цепи от t ко всем другим вершинам из $G^{\mu}(\xi)$ (например, с использованием алгоритма из 2.2.1 гл. 8), то все циклы отрицательной стоимости в $G^{\mu}(\xi)$ могут быть обнаружены.

Используя такой подход при организации шага 3 основного алгоритма потока минимальной стоимости из разд. 5.1, Беннингтон [2] опубликовал вычислительные результаты, показывающие, что этот алгоритм не хуже алгоритма «беспорядка» [12, 4, 7]. Применение метода наименьших квадратов к оценке требуемого времени t вычисления по тестам, использующим случайно

порожденные графы с n вершинами и m дугами, дало такой результат:

$t = -2{,}3 + 0{,}0113n + 0{,}00166m$ (время в секундах на машине ИБМ 360/75).

Граф с 200 вершинами и 5000 дугами требует приблизительно 8 с машинного времени.

5.1.4. Потоки минимальной стоимости в графах с выпуклыми функциями стоимости

До сих пор предполагалось, что стоимость единицы потока ξ_{ij} по дуге (x_i, x_j) есть c_{ij} — постоянная величина. Однако это не является необходимым предположением и, как показали Ху [17] и Клейн [20], алгоритм, описанный в 5.1.1, можно легко применить и для нахождения потоков минимальной стоимости в графах, стоимости дуг которых являются выпуклыми кусочно-линейными функциями потока.

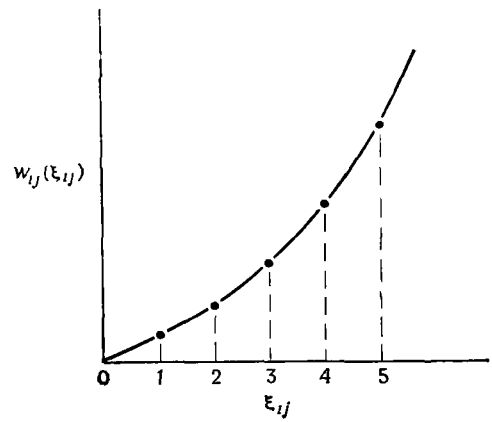


Рис. 11.15. Выпуклая функция стоимости дуг.

Если допустить, что стоимость потока ξ_{ij} по дуге (x_i, x_j) задается выпуклой функцией $w_{ij}(\xi_{ij})$, изображенной на рисунке 11.15, то для любого потока ξ мы можем определить инкрементальный граф $G^{\mu}(\xi) = (X^{\mu}, A_1^{\mu} \cup A_2^{\mu})$ точно так же, как это делалось в разд. 2.3, за исключением того, что теперь никогда не нужно вычислять пропускные способности дуг q_{ij}^{μ} и q_{ji}^{μ}, а стоимости дуг таковы:

для дуги $(x_i^{\mu}, x_j^{\mu}) \in A_1^{\mu}$ положим

$$c_{ij}^{\mu} = w_{ij}(\xi_{ij} + 1) - w_{ij}(\xi_{ij}),$$

для дуги $(x_j^\mu, x_i^\mu) \in A_2^\mu$ положим

$$c_{ji}^\mu = -[w_{ij}(\xi_{ij}) - w_{ij}(\xi_{ij} - 1)],$$

считая, что $w_{ij}(0) = 0$.

Теперь мы можем непосредственно использовать алгоритм из разд. 5.1.1, за исключением того, что δ — поток, посылаемый по любому циклу Ф отрицательной стоимости, который можно найти в $G^\mu(\xi)$,— всегда берется равным 1. Заметим, что все инкрементальные стоимости были определены так, чтобы они приводили к увеличению или уменьшению значения потока на одну единицу. Заметим также, что при использовании только «единичных» преобразований потоков (т. е. при уменьшении или увеличении значения потока только на единицу) нет необходимости вычислять инкрементальные пропускные способности дуг в $G^\mu(\xi)$, так как эти величины нужны лишь для вычисления δ.

Если при задании функции стоимости не требуется большой точности, то $w_{ij}(\xi_{ij})$ можно аппроксимировать кусочно-линейной функцией. Тогда ограничение, что δ выбирается равным единице потока, можно ослабить и достигнуть более быстрой сходимости к потоку минимальной стоимости. Инкрементальные стоимости каждой дуги могут быть теперь определены как угловые коэффициенты соответствующих линейных участков функций стоимости, а именно для данной дуги выбирается участок, отвечающий текущему значению потока на этой дуге. Затем можно вычислить пропускные способности, чтобы убедиться, что никакое увеличение или уменьшение потока δ не даст новый поток, выходящий за пределы применимости текущего линейного участка, аппроксимирующего функцию стоимости.

5.2. Алгоритм, основанный на цепи наименьшей стоимости

Алгоритм из разд. 5.1 начинает работу с произвольного потока ξ со значением v, после чего происходит постепенное улучшение потока до тех пор, пока не будет получен поток ξ^* минимальной стоимости. В терминологии математического программирования такой алгоритм следовало бы назвать *основным* алгоритмом, так как его первая цель состоит в получении допустимого потока, а затем поток улучшается с сохранением допустимости. Альтернативным подходом является *двойственный* метод, в котором для решения этой задачи начинают с построения потока ξ^* с минимальной стоимостью, имеющего некоторое значение $v_0 < v$, а затем добавляют в граф некоторый дополнительный поток, чтобы получить новый поток с минимальной стоимостью, имеющий значение $v_1 > v_0$, и т. д. пока не будет построен поток с минимальной стоимостью со значением v. В отношении этого метода следует

заметить, что начальный поток минимальной стоимости всегда имеется, так как нулевой поток является потоком минимальной стоимости со значением 0.

Алгоритм, который мы собираемся описать, основан на вычислении кратчайшей цепи и следующей теореме.

Теорема 6. *Если* ξ — *поток минимальной стоимости в графе* G *со значением* v *и* P^* — *кратчайшая цепь* (*наименьшей стоимости*) *от* s *к* t *в инкрементальном графе* $G^\mu(\xi)$, *то* $\xi + 1 \circ (P^*)$ *является потоком минимальной стоимости со значением* $v + 1$.

Доказательство теоремы очень просто — надо показать, что если $G^\mu(\xi)$ не содержит никакого цикла отрицательной стоимости, то $G^\mu(\xi + 1 \circ (P^*))$ также не содержит таких циклов. После этого все следует непосредственно из теоремы 5.

5.2.1. Описание алгоритма

Шаг 1. Начать с нулевых потоков на дугах и с потока, имеющего нулевое значение.

Шаг 2. Построить инкрементальный граф $G^\mu(\xi) = (X^\mu, A_1^\mu \cup A_2^\mu)$, как это описано в разд. 2.3, и взять следующие стоимости дуг:

для любой дуги $(x_i^\mu, x_j^\mu) \in A_1^\mu$ положить $c_{ij}^\mu = c_{ij}$,

для любой дуги $(x_j^\mu, x_i^\mu) \in A_2^\mu$ положить $c_{ji}^\mu = -c_{ij}$.

Шаг 3. Найти в графе $G^\mu(\xi)$ кратчайшую цепь P^* от вершины s к вершине t.

Шаг 4. Найти наибольшую величину δ того потока, который можно послать вдоль P^* не изменяя структуру графа $G^\mu(\xi)$. Величина δ равна

$$\delta = \min_{(x_i^\mu, x_j^\mu) \text{ в } P^*} [q_{ij}^\mu].$$

Шаг 5. (I) Если значение потока $\xi + \delta \circ (P^*)$ меньше, чем требуемая величина v, произвести замену $\xi \leftarrow \xi + \delta \circ (P^*)$ и вернуться к шагу 2.

(II) Если значение потока $\xi + \delta \circ (P^*)$ равно v, остановиться. $\xi + \delta \circ (P^*)$ является требуемым потоком минимальной стоимости.

(III) Если значение потока $\xi + \delta \circ (P^*) = h > v$, остановиться. Требуемым потоком минимальной стоимости будет $\xi + (\delta - h + v) \circ (P^*)$.

Здесь важно отметить, что граф $G^\mu(\xi)$ на шаге 2 содержит как дуги положительной стоимости, так и отрицательной, и поэтому алгоритм кратчайшей цепи, используемый на шаге 3, не может быть алгоритмом Дейкстры, а должен быть (менее эффективным)

алгоритмом, описанным в разд. 2.2.1 гл. 8. Однако Хаувер, Эдмондс и Карп предложили недавно метод, позволяющий обойти вышеуказанную трудность и использовать более эффективные алгоритмы кратчайшей цепи.

5.2.2. **Пример.** Снова вычислим поток минимальной стоимости со значением 20 для графа, изображенного на рис. 11.13. Вначале берем $\xi = 0$. Инкрементальный граф изображен на рис. 11.13. Кратчайшей цепью в графе будет $P_1^* = (x_1, x_2, x_4, x_3, x_7)$ со значением 22, а δ оказывается равным 12.

Поток ξ переходит в $12 \circ (P_1^*)$. Кратчайшей цепью в соответствующем инкрементальном графе будет $P_2^* = (x_1, x_2, x_4, x_6, x_7)$ со стоимостью 23 и δ равным 4.

Поток становится равным $12 \circ (P_1^*) + 4 \circ (P_2^*)$. Кратчайшей цепью в $G^{\mu}(\xi)$ будет $P_3^* = (x_1, x_4, x_6, x_7)$ со стоимостью 25 и $\delta = 1$.

Поток становится равным $12 \circ (P_1^*) + 4 \circ (P_2^*)$. Кратчайшей цепью в $G^{\mu}(\xi)$ будет $P_3^* = (x_1, x_4, x_6, x_7)$ со стоимостью 31, и вдоль этой цепи можно послать 3 оставшиеся единицы потока для получения потока минимальной стоимости, изображенного на рис. 11.14 л.

6. Потоки в графах с выигрышами

До сих пор в этой главе делалось предположение, что поток, выходящий из дуги, является таким же, как и поток, входящий в дугу. Однако в ряде практических ситуаций это допущение не выполняется. Например, в трубопроводе, по которому перекачивается жидкость, или в электрической сети могут быть потери, уменьшающие поток по дуге. В системах транспортировки может происходить порча груза, которая также приводит к уменьшению потока по дуге, представляющей маршрут. В технологических процессах, которые могут быть изображены графами с дугами, соответствующими операциям, ценность материала, покидающего дугу, больше, чем ценность материала, входящего в дугу, так как в дуге имеется выигрыш, получаемый за счет «прибавочной стоимости» операции. В задачах обмена (например, при продаже и покупке валюты) выигрыши дуг представляют обменный курс входного потока (измеряемый в одной валюте) по отношению к выходному потоку (измеряемому в другой валюте).

В этом разделе мы рассмотрим задачу нахождения максимального потока (от s к t) в графе с произвольными неотрицательными выигрышами g_{ij} и пропускными способностями q_{ij}, приписанными дугам (x_i, x_j) графа G. Эта задача аналогична задаче о максимальном потоке (от s к t), обсуждавшейся в разд. 2, хотя теперь входной

и выходной потоки связаны между собой только через «посредство» графа G, способного как «породить», так и «поглотить» поток.

Если через ξ^e_{ij} обозначить поток, входящий в дугу (x_i, x_j), а через ξ^o_{ij} — поток, выходящий из этой дуги, то

$$\xi^o_{ij} = g_{ij}\xi^e_{ij}. \tag{11.12}$$

Предположим далее, что пропускные способности дуг согласованы только с входными потоками, т. е. для любой дуги

$$\xi^e_{ij} \leqslant q_{ij} \tag{11.13}$$

независимо от величины ξ^o_{ij}. Обозначим чистый входной поток [1]) в s через v_s, а чистый выходной поток [2]) в t через v_t. Поток $\Xi = (\xi^e, \xi^o)$ является допустимым, если он удовлетворяет условию непрерывности потока во всех вершинах, т. е.

$$\sum_{x_j \in \Gamma(x_i)} \xi^e_{ij} - \sum_{x_j \in \Gamma^{-1}(x_i)} \xi^o_{ji} = \begin{cases} v_s & \text{для } x_i = s, \\ -v_t & \text{для } x_i = t, \\ 0 & \text{для } x_i \neq s,\ t, \end{cases} \tag{11.14}$$

а также условиям (11.12) и (11.13) для каждой дуги (x_i, x_j) из G.

Введем теперь два определения.

Определение. Допустимый поток $\hat{\Xi} = (\hat{\xi}^e, \hat{\xi}^o)$ называется *максимальным*, если он имеет наибольшее значение v_t (скажем, $\hat{v}_t$) среди всех допустимых потоков.

Определение. Допустимый поток $\tilde{\Xi} = (\tilde{\xi}^e, \tilde{\xi}^o)$ называется *оптимальным*, если для любого другого допустимого потока Ξ

$$\text{или } v_t \leqslant \tilde{v}_t, \text{ когда } v_s = \tilde{v}_s,$$
$$\text{или } v_s \geqslant \tilde{v}_s, \text{ когда } v_t = \tilde{v}_t,$$

где $\tilde{v}_s$ и $\tilde{v}_t$ являются соответственно чистыми потоками в источнике и стоке потока $\tilde{\Xi}$.

Эти последние условия согласуются с интуитивным пониманием оптимальности, а именно заданные $\tilde{v}_s$, $\tilde{v}_t$ являются или наибольшими, или наименьшими возможными.

Определение. Допустимый поток $\hat{\tilde{\Xi}}$ называется *оптимально-максимальным*, если он является как оптимальным, так и максимальным потоком.

Проиллюстрируем введенные понятия на примере. Рассмотрим граф на рис. 11.16 а, где первое число у дуги задает ее пропускную способность, а второе — выигрыш. Тогда

[1]) То есть поток, «порождаемый» в самой вершине s.— *Прим. ред.*
[2]) То есть поток, «поглощаемый» в самой вершине t.— *Прим. ред.*

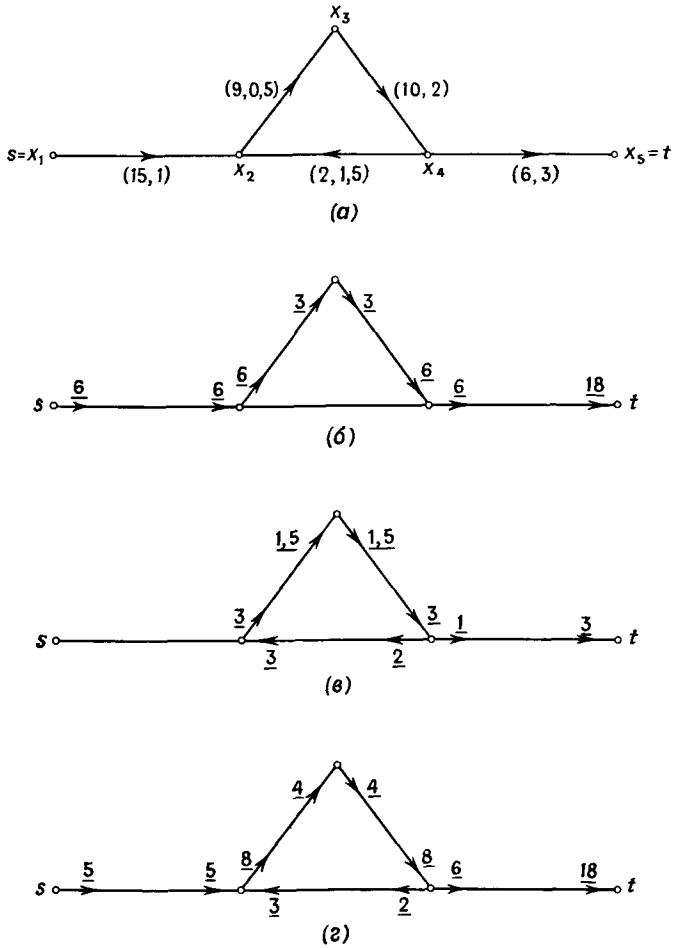


Рис. 11.16. (а) Граф G: первая пометка — пропускная способность дуги, вторая — выигрыш дуги. (б) Максимальный поток. (в) Оптимальный поток. (г) Оптимально-максимальный поток.

(I) поток, изображенный на рис. 11.16 б, является максимальным потоком со значениями $\hat{v}_t = 18$ и $\hat{v}_s = 6$;

(II) поток, изображенный на рис. 11.16 в, является оптимальным потоком со значениями $\tilde{v}_t = 3$ и $\tilde{v}_s = 0$ (заметим, что поток порожден циркуляцией по циклу, у которого полный выигрыш больше чем 1; кроме того, $\tilde{v}_t < \hat{v}_t$ и, следовательно,

поток не является максимальным, но $\tilde{v}_t$ представляет собой максимум, который может быть получен при $v_s = 0$);

(III) поток, изображенный на рис. 11.16 (г), является оптимально-максимальным, имеющим $\hat{\tilde{v}}_t = \hat{v}_t$, но $\hat{\tilde{v}}_s = 5 < \hat{v}_s = 6$ и, как будет показано ниже, наименьшее возможное значение v_s, дающее выходной поток со значением 18, равно 5.

6.1. Аугментальные маршруты

Рассмотрим граф с выигрышами, в котором существует допустимый поток $\Xi = (\xi^e, \xi^o)$. Обозначим этот граф через $G(\Xi)$. Маршрут (не обязательно цепь от вершины x_{i_1} к вершине x_{i_l} называется *аугментальным маршрутом* от x_{i_1} к x_{i_l}, если в графе $G(\Xi)$ по этому маршруту можно «послать» поток от x_{i_1} к x_{i_l}. (Это аналогично определению аугментальной цепи потока (от s к t) в задаче о максимальном потоке (от s к t).) Если маршрут задан последовательностью вершин $x_{i_1}, x_{i_2}, \ldots, x_{i_l}$, то пусть F — множество всех «прямых» дуг в нем (т. е. дуг $(x_{i_p}, x_{i_{p+1}})$ с $x_{i_{p+1}} \in \Gamma(x_{i_p})$) и B — множество всех «обратных» дуг (т. е. дуг, для которых $x_{i_p} \in \Gamma(x_{i_{p+1}})$). Маршрут является аугментальным, если $\xi^e_{ij} < q_{ij}$ для каждой прямой дуги $(x_i, x_j) \in F$ и $\xi^e_{ji} > 0$ для каждой обратной дуги $(x_j, x_i) \in B$.

А. **Выигрыш маршрута.** *Выигрыш* маршрута $S = (x_{i_1}, \ldots x_{i_l})$ определяется так:

$$g(S) = \prod_{(x_i, x_j) \in F} g_{ij} \cdot \prod_{(x_j, x_i) \in B} 1/g_{ji}. \qquad (11.15)$$

Б. **Пропускная способность маршрута.**

Инкрементальная пропускная способность аугментального маршрута S равна значению такого максимального входного потока в вершине x_{i_1}, который может быть послан по данному маршруту к вершине x_{i_l} и либо насыщает поток в некоторой прямой дуге из S, либо обращает в нуль поток в некоторой обратной дуге маршрута S.

Если S — цепь и в x_{i_1} входит поток величины δ, то входной поток (в вершине x_{i_p}) на дуге $(x_{i_p}, x_{i_{p+1}})$ из S равен

$$\delta_{i_p i_{p+1}} = \delta \cdot \prod_{(x_i, x_j) \in F_p} g_{ij} \cdot \prod_{(x_j, x_i) \in B_p} 1/g_{ji} \equiv \delta \cdot g(S_p), \qquad (11.16)$$

где F_p и B_p — множества соответственно прямых и обратных дуг из подцепи $S_p = (x_{i_1}, x_{i_2}, \ldots, x_{i_p})$ и $g(S_p)$ — выигрыш цепи S_p.

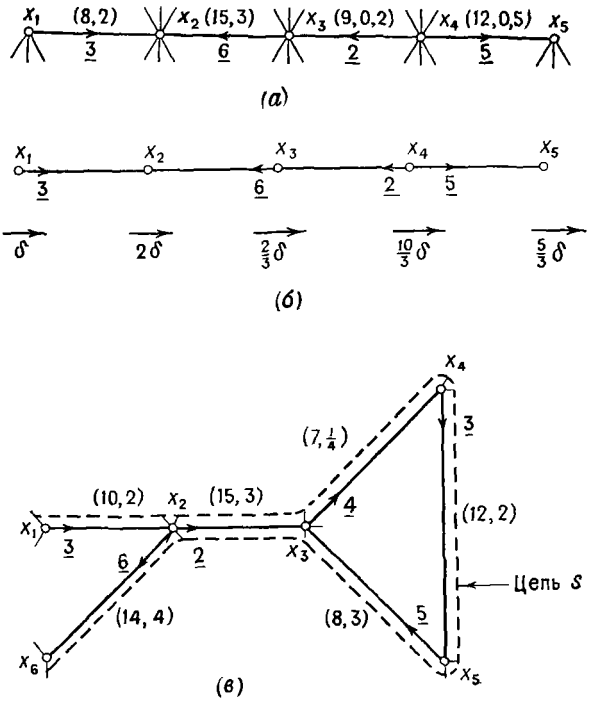


Рис. 11.17. (а) Цепь S с начальными входными потоками, показанными подчеркнутыми числами: первая пометка в скобках — пропускная способность дуги, вторая — выигрыш дуги. (б) Цепь S с введенным дополнительным потоком. (в) Маршрут S с начальным входным потоком, показанным подчеркнутыми числами: первая пометка в скобках — пропускная способность дуги, вторая — выигрыш дуги. (г) Маршрут S с введенным дополнительным потоком.

Если $(x_{i_p}, x_{i_{p+1}})$ — прямая дуга из S, несущая поток $\xi^e_{i_p i_{p+1}}$, то эта дуга будет насыщена, если $\xi^e_{i_p i_{p+1}} + \delta_{i_p i_{p+1}} = q_{i_p i_{p+1}}$. Если же поток $\xi^e_{i_{p+1} i_p}$ несет обратная дуга, то ее поток станет равным нулю, когда $\delta_{i_p i_{p+1}} = \xi^o_{i_{p+1} i_p}$, т. е. когда $\delta_{i_p i_{p+1}} = g_{i_{p+1} i_p}\, \xi^e_{i_{p+1} i_p}$. Инкрементальная пропускная способность цепи S дается выражением

$$q(S) = \min\left[\min_{(x_{i_p}, x_{i_{p+1}})\in F}\left\{\frac{q_{i_p i_{p+1}} - \xi^e_{i_p i_{p+1}}}{g(S_p)}\right\},\right.$$

$$\left.\min_{(x_{i_{p+1}}, x_{i_p})\in B}\left\{\frac{g_{i_{p+1} i_p}\, \xi^e_{i_{p+1} i_p}}{g(S_p)}\right\}\right]. \qquad (11.17)$$

Если маршрут S не является цепью и, следовательно, некоторые дуги встречаются более чем один раз, то и в этом случае легко получить выражение, аналогичное (11.17).

Для цепи S на рис. 11.17 (а) и заданного начального потока добавление потока δ в вершине x_1 приводит к потоку, изображенному на рис. 11.17 (б). Максимальное значение δ, оставляющее поток допустимым, равно тогда $q(S) = 0{,}6$. Для этого значения поток по дуге (x_4, x_3) сводится к нулю, т. е. S перестает быть аугментальной цепью в новом графе $G(\Xi)$. Если бы начальный поток, входящий в дугу (x_4, x_3), был равен, скажем, 8 вместо 2, то максимальное допустимое значение δ было бы тогда 2,1 — значение, при котором дуга (x_4, x_3) из S была бы насыщена.

Для маршрута S, изображенного на рис. 11.17 (в) и заданного начального потока добавление потока δ в вершине x_1 даст поток, показанный на рис. 11.17 (г). Инкрементальная пропускная способность этого маршрута равна 0,5 — значение, при котором дуга (x_3, x_4) становится насыщенной.

6.2. Активные циклы

Так как цикл [1]) можно рассматривать как маршрут с совпадающими начальной и конечной вершинами, то выигрыш цикла и его пропускная способность могут быть определены так же, как и для маршрута. Цикл Ф называется *активным* в графе $G(\Xi)$ по отношению к вершине t, если

(I) его выигрыш больше единицы,

(II) его инкрементальная пропускная способность отлична от нуля,

(III) на Ф имеется некоторая вершина x_i, для которой существует аугментальный маршрут от x_i к t.

[1]) Точнее — замкнутый маршрут.— *Прим. ред.*

Активный по отношению к начальной вершине s цикл может быть определен аналогично. Из вышеприведенного определения ясно, что с помощью циркуляции потока по активному циклу можно создать дополнительный поток (по свойству (I)) и передать его в вершину t (по свойству (III)). В вышеприведенном примере на рис. 11.16(в) поток порождался циркуляцией по активному циклу (x_4, x_2, x_3, x_4).

Активный по отношению к t цикл называется *дезактивированным*, если добавление дополнительного потока в G осуществлено так, что или инкрементальная пропускная способность цикла уменьшилась до нуля, или пропускные способности всех аугментальных цепей от какой-либо вершины на цикле до вершины t уменьшились до нуля (т. е. в новом графе $G(\Xi)$ не может быть сделано больше никакого добавления ненулевого дополнительного потока). В [13] и [22] доказаны следующие утверждения.

Теорема 7. *Поток* $\widetilde{\Xi}$ *является оптимальным тогда и только тогда, когда граф* $G(\widetilde{\Xi})$ *не содержит никаких активных циклов по отношению к* s *или* t.

Теорема 8. *Пусть* $\widetilde{\Xi}$ *является оптимальным потоком. Тогда если поток увеличить по какой-либо (от* s *к* t*) аугментальной цепи с наивысшим выигрышем, то вновь полученный поток также будет оптимальным.*

Теорема 9. *Поток* $\widehat{\widetilde{\Xi}}$ *будет оптимально-максимальным, если некоторый разрез* $(X_0 \to X - X_0)$*, отделяющий* s *от* t*, является насыщенным и если не существует никакого активного цикла по отношению к* t*, все вершины которого лежат в* $X - X_0$.

6.3. Алгоритм оптимального потока для графов с выигрышами

Из трех вышеприведенных теорем сразу же вытекает алгоритм оптимальности, описанный ниже.

Шаг 1. Начать с произвольного допустимого потока Ξ в G (допустим, например, нулевой по каждой дуге поток).

Шаг 2. Найти в $G(\Xi)$ активный цикл Φ по отношению к t.

Шаг 3. Пусть x_i — такая вершина в Φ, что аугментальный маршрут начинается в x_i и кончается в t. (Заметим, что x_i может совпадать с t.) Начиная с вершины x_i, устроить циркуляцию потока δ по активному циклу и передать избыток потока из x_i в t по аугментальному маршруту. Выбрать δ так, чтобы (вместе с существующим потоком Ξ) инкрементальная пропускная способность цикла или аугментального маршрута уменьшилась до нуля.

Шаг 4. Обновить поток Ξ и возвращаться к шагу 2 до тех пор, пока все активные циклы не будут дезактивированы и больше не будет существовать никаких активных циклов; после чего перейти к шагу 5. (На этом этапе поток Ξ будет оптимальным потоком $\widetilde{\Xi}$ с $v_s = 0$.)

Шаг 5. Найти аугментальный маршрут от s к t с наибольшим выигрышем. Посылать поток по этому маршруту до тех пор, пока инкрементальная пропускная способность не уменьшится до нуля.

Шаг 6. Обновить поток Ξ и повторять шаг 5 до тех пор, пока или чистый выходной поток v_t не примет требуемое значение (оптимального потока), или больше не будет существовать никаких аугментальных маршрутов; тогда полученный поток будет оптимально-максимальным потоком $\widehat{\widetilde{\Xi}}$.

Шаги вышеприведенного алгоритма очень просты, а сам метод эффективен. Шаги 2 и 5 можно выполнять и с помощью алгоритма кратчайшей цепи. А именно определим инкрементальный граф $G^{\mu}(\Xi)$, соответствующий графу $G(\Xi)$, с множеством вершин $X^{\mu} = X$ и множеством дуг A^{μ}, таким, что

$$x_i^{\mu}, x_j^{\mu} \in A^{\mu}, \quad \text{если} \quad 0 \leqslant \xi_{ij}^{e} < q_{ij},$$

причем инкрементальная «стоимость»

$$c_{ij}^{\mu} = -\log(g_{ij})$$

и пропускная способность

$$q_{ij}^{\mu} = q_{ij} - \xi_{ij}^{e},$$

и

$$(x_j^{\mu}, x_i^{\mu}) \in A^{\mu}, \quad \text{если} \quad 0 < \xi_{ij}^{e} \leqslant q_{ij},$$

инкрементальная «стоимость»

$$c_{ji}^{\mu} = -\log(1/g_{ij})$$

и пропускная способность

$$q_{ji}^{\mu} = \xi_{ij}^{e} \cdot g_{ij}.$$

Цикл Ф в графе $G(\Xi)$, имеющий выигрыш, больший чем единица, соответствует тогда циклу с отрицательной «стоимостью» в графе $G^{\mu}(\Xi)$. В этом можно убедиться, прологарифмировав обе части соотношения (11.15), в котором в качестве маршрута S надо взять цикл Ф. Тогда получим

$$\log[g(\Phi)] = \sum_{(x_i, x_j) \in F} \log(g_{ij}) + \sum_{(x_j, x_i) \in B} \log(1/g_{ji})$$

$$= -\left[\sum_{F} c_{ij}^{\mu} + \sum_{B} c_{ij}^{\mu}\right].$$

где F и B, как и прежде,— множества прямых и обратных дуг цикла Ф.

Если $g(\Phi) > 1$, то $\log[g(\Phi)] > 0$, откуда следует, что стоимость цикла Ф в $G^{\mu}(\Xi)$, т. е. $\sum\limits_{F} c_{ij}^{\mu} + \sum\limits_{B} c_{ij}^{\mu}$, должна быть отрицательной. Обратно, если $g(\Phi) \leqslant 1$, то стоимость цикла Ф в $C(\Xi)$ неотрицательна.

Таким образом, активные циклы в $G(\Xi)$ — на шаге 2 вышеописанного алгоритма — могут быть определены посредством нахождения всех кратчайших (наименьшей «стоимости») цепей, идущих из вершин графа $G^{\mu}(\Xi)$ к конечной вершине t (для чего можно воспользоваться методом из разд. 2.2.1 гл. 8, как это было объяснено в разд. 4 настоящей главы), и проверки циклов на отрицательную стоимость. Если, однако, поток δ, циркулирующий по активному циклу Ф на шаге 3 и передаваемый по аугментальному маршруту к t, не уменьшает до нуля пропускную способность самого цикла Ф, а вместо этого уменьшает до нуля пропускную способность аугментального маршрута, то при следующем выполнении шага 2 нужно только найти другой аугментальный маршрут, ведущий из вершины цикла Ф в вершину t. Такой маршрут снова сделает цикл активным и можно будет перейти к шагу 3 и т. д. вплоть до того момента, когда цикл Ф дезактивируется. Тогда на шаге 2 ищется некоторый другой активный цикл.

Шаг 5 вышеописанного алгоритма также можно выполнить с помощью метода разд. 2.2.1 гл. 8, так как он состоит просто в вычислении кратчайшей цепи от s к t в графе $G^{\mu}(\Xi)$. Этот граф «освобождается» от циклов отрицательной стоимости, как только выполнены четыре предыдущие шага алгоритма.

6.4. Пример

Мы хотим найти оптимально-максимальный поток от вершины x_1 к вершине x_8 в графе на рис. 11.18, где первое число у дуги

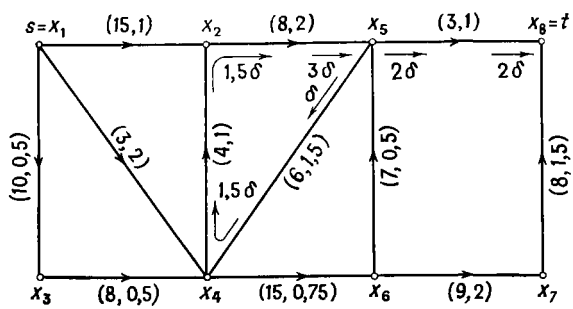


Рис. 11.18. Граф из примера 6.4. Первая пометка — пропускная способность дуги, вторая — выигрыш дуги.

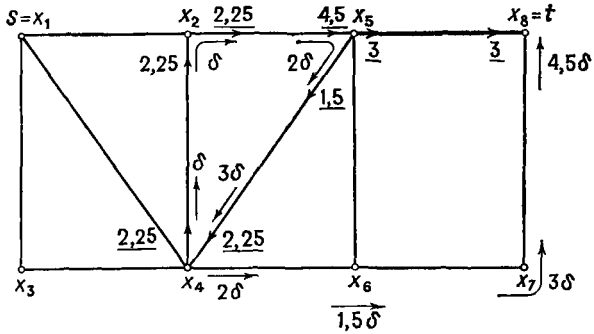


Рис. 11.19а. Поток после 1-го этапа.
— насыщенная дуга.

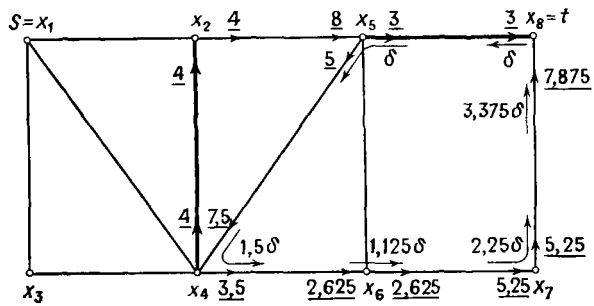


Рис. 11.19б. Поток после 2-го этапа.
— насыщенные дуги.

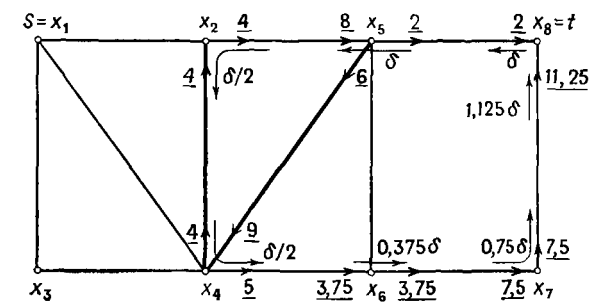


Рис. 11.19в. Поток после 3-го этапа.
— насыщенные дуги.

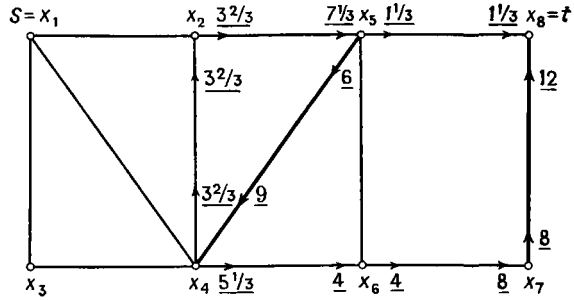


Рис. 11.19. г. Оптимальный поток.
— насыщенные дуги.

является ее пропускной способностью, а второе — ее выигрышем.

Начиная с $\Xi = (0, 0)$ находим первый активный цикл $\Phi = (x_5, x_4, x_2, x_5)$ с аугментальным маршрутом (x_5, x_8). Циркуляция потока δ (где δ — входной поток дуги (x_5, x_4)) по этому маршруту, как показано на рис. 11.18, дает максимальный допустимый поток со значением $\delta = 1{,}5$, и для этого значения получающийся поток показан (подчеркнутыми числами) на рис. 11.19а, причем дуга (x_5, x_8) насыщена. Пропускная способность самого цикла не уменьшается до нуля, и поэтому при следующей итерации, когда алгоритм кратчайшей цепи (действующий от всех других вершин графа $G^{\mu}(\Xi)$ к вершине t) помечает вершину x_4 из Φ, цикл Φ снова становится активным с новой аугментальной цепью (x_4, x_6, x_7, x_8). Посылая поток δ по циклу и передавая его по аугментальной цепи к t, как показано на рис. 11.19а, получаем поток, изображенный на рис. 11.19б подчеркнутыми числами.

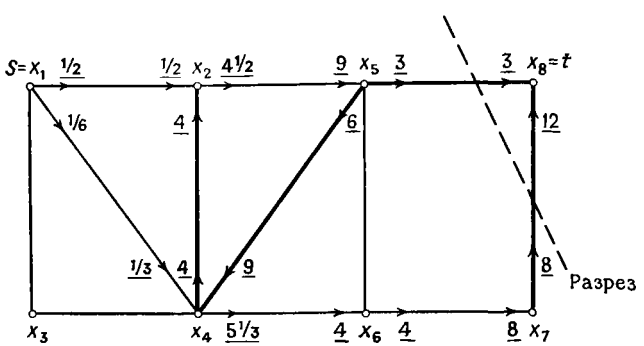


Рис. 11.19д. Оптимально-максимальный поток.
— насыщенные дуги.

Новый активный цикл $(x_8, x_5, x_4, x_6, x_7, x_8)$ определяется на шаге 2, и циркуляция потока δ по этому циклу, как показано на рис. 11.19(б), дает поток, изображенный подчеркнутыми числами на рис. 11.19(в). Наконец, определяется активный цикл $(x_8, x_5, x_2, x_4, x_6, x_7, x_8)$ и его дезактивация с помощью циркуляции потока приводит к потоку, изображенному на рис. 11.19г. Последний будет оптимальным потоком, так как больше не удается обнаружить никаких активных циклов.

Аугментальный маршрут (от s к t) с максимальным выигрышем есть $(x_1, x_4, x_2, x_5, x_8)$ с пропускной способностью $1/6$. Как только этот маршрут будет насыщен (дуга (x_4, x_2) уже насыщена), аугментальным маршрутом со следующим наибольшим выигрышем будет (x_1, x_2, x_5, x_8), и он после насыщения даст поток на рис. 11.19д. Этот поток будет требуемым оптимально-максимальным потоком, так как больше нельзя найти никаких аугментальных цепей (от s к t).

6.5. Графы с выигрышами в вершинах

Задача о максимальном потоке (от s к t) для случая, когда пропускными способностями обладают вершины, была решена в разд. 3.2 с помощью простой замены каждой вершины x_j парой вершин x_j^+ и x_j^-. Аналогичным образом, если вершинам графа G приписаны выигрыши и, следовательно, условия (11.14) непрерывности потока не выполняются, то мы можем образовать другой граф G_0, заменяя каждую вершину x_j парой вершин x_j^+ и x_j^- с дугой (x_j^+, x_j^-) между ними. Выигрыш этой дуги считается равным выигрышу вершины x_j. Для дуги (x_i, x_j) из G мы тогда имеем дугу (x_i^-, x_j^+) в G_0, а для дуги (x_j, x_k) из G — дугу (x_j^-, x_k^+) в G_0. Задача после этого сводится к нахождению оптимальных потоков в графе G_0, в котором выигрыши приписаны только дугам.

7. Задачи

1. Найти максимальный поток ξ^* от x_1 к x_7 в графе, изображенном на рис. 11.20, где пропускные способности дуг указаны стоящими около них числами.

2. Начертить инкрементальный граф $G^{\mu}(\xi)$ для примера из задачи 1.

3. Разложить поток, найденный в задаче 1, на элементарные потоки (от s к t) по цепям и циклам.

4. Пусть в графе, изображенном на рис. 11.20, потоки по дугам (x_2, x_3), (x_5, x_4) и (x_2, x_5) ограничены снизу значениями $r_{23} = 4$, $r_{54} = 7$ и $r_{25} = 3$ соответственно. Найти максимальный поток от x_1 к x_7 или показать, что такой поток не существует.

5. Доказать теорему 3 из разд. 3.3.

6. Найти максимальные потоки между всеми парами вершин для графа G, изображенного на рис. 11.21, и начертить потоково-эквивалентное дерево. Определить разрезы в G, соответствующие этому потоково-эквивалентному дереву. Найти два других потоково-эквивалентных дерева.

7. Найти максимальный поток с минимальной стоимостью от x_1 к x_8 в графе на рис. 11.22, если первое число у дуги равно ее пропускной способности, а второе — стоимости.

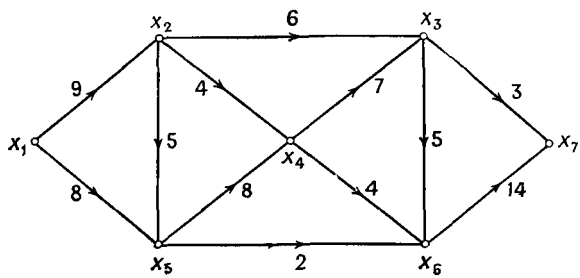


Рис. 11.20.

8. Какие упрощения могут быть сделаны в алгоритме потока минимальной стоимости в случае, когда требуемое значение потока является максимально возможным?

9. В алгоритме максимального потока (от s к t) из разд. 2.1 среди многих возможностей, возникающих при выборе аугментальных цепей, две таковы:

(I) выбирается аугментальная цепь с наименьшим числом дуг,

(II) выбирается та цепь, которая позволяет послать от s к t аугментальный поток с максимально возможным значением δ.

Какая из этих двух возможностей предпочтительнее в вычислительном отношении и как зависят они от величины пропускной способности дуг? (см. [9]).

10. Доказать теорему 6 из разд. 5.2.

11. Доказать теорему 7 из разд. 6.2.

12. В графе, изображенном на рис. 11.23, найти оптимальный, максимальный и оптимально-максимальный потоки, если первое число, стоящее у дуги, равно ее пропускной способности, а второе — ее выигрышу.

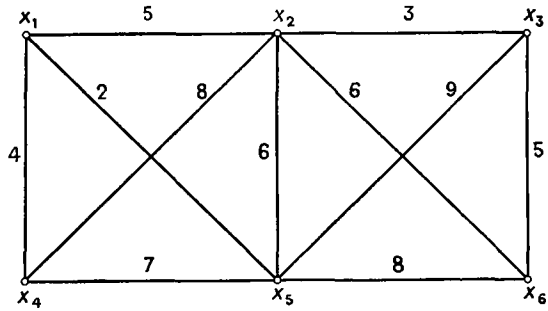


Рис. 11.21.

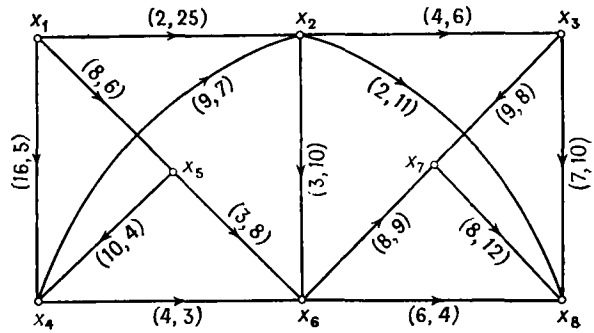


Рис. 11.22.

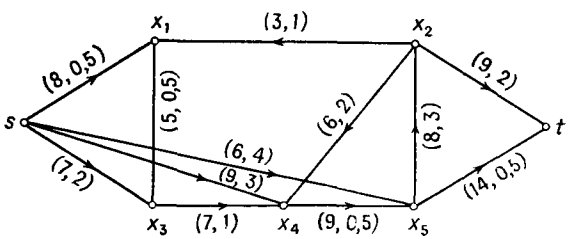


Рис. 11.23.

8. Список литературы

1. Beale E. M. L. (1959), An algorithm for solving the transportation problem when the shipping cost over each route is convex, *Nav. Res. Log. Quart.*, 6, p. 43.
2. Bennington G. E. (1972), An efficient minimal cost flow algorithm, Report N° 75, Department of Industrical Engineering, North Carolina State University at Raleigh.
3. Boldyreff A. W. (1955), Determination of the maximal steady flow of traffic through a railroad network, *Ops. Res.*, 3, p. 443.
4. Bray J. A., Wizgall C. (1968), Algorithm 336 NETFLOW, *Comm. of ACM*, 11, p. 631.
5. Басакер Р., Саати Т., Конечные графы и сети, М., Наука, 1974.
6. Busacker R. G., Gowen P. J. (1961), A procedure for determining a family of minimal-cost network flow patterns, Operations Research Office, Technical paper 15.
7. Clasen R. J. (1967), RS-OKF3, SHARE IBM Users group SDA 3536.
8. Dantzig G. B., Fulkerson D. R. (1956), On the max-flow min-cut theorem of networks, Linear inequalities and related systems, *Ann. Math. Studies*, 38, p. 215.
9. Edmonds J., Karp R. M. (1972), Theoretical improvements in algorithmic efficiency for network flow problems, *J. of ACM*, 19, p. 248.
10. Elias P., Feinstein A., Shannon C. E. (1956), A note on the maximum flow through a network, *IRE Trans.*, IT-2, p. 117.
11. Ford L. R., Fulkerson D. R. (1956), Maximal flow through a network, *Can. J. of Math.*, 18, 399.
12. Ford L. R., Fulkerson D. R.(1962), Flows in networks, Princeton University Press, Princeton.
13. Frank H., Frisch I. T. (1971), Communication, transmission and transportation networks, Addison Wesley, Reading, Massachusetts.
14. Gomory R. E., Hu T. C. (1964), Synthesis of a communication network, *J. of SIAM (Appl. Math)*, 12, p. 348.
15. Grigoriadis M. D., White W. W. (1972), A partitioning algorithm for the multicommodity network flow problem, *Math. Progr.*, 3, p. 157.
16. Hu T. C. (1969), Integer programming and network flows, Addison Wesley, Reading, Massachusetts.
17. Hu T. C. (1966), Minimum cost flow in convex cost networks, *Nav. Res. Log. Quart.*, 13, p. 1.
18. Jewell W. S. (1968), Multicommodity network solutions, Theorie de graphes, Dunod, Paris, p. 183.
19. Johnson E. (1966), Networks and basic solutions, *Ops. Res.*, 14, p. 619.
20. Klein M. (1967), A primal method for minimal cost flows with applications to the assignment and transportation problems, *Man. Sci.*, 14, p. 205.
21. Maurras J. F. (1972), Optimization of the flow through networks with gains, *Math. Prog.*, 3, p. 135.
22. Onaga K. (1967), Optimum flows in general communication networks, *J. of the Franklin Institute*, 283, p. 308.
23. Potts R. B., Oliver R. M. (1972), Flows in transportation networks, Academic Press, New York.
24. Sakarovitch M. (1966), The multicommodity maximum flow problem, Report ORC-66-25, Operations Research Centre, Univ. of California, Berkeley.
25. Siagal R. (1967), Multicommodity flows in directed networks, Report ORC-67-38, Operations Research Centre, Univ. of California, Berkeley.
26. Tomlin J. A. (1966), Minimum cost multicommodity network flows, *Ops. Res.*, 14, p. 45.
27. Wollmer R. D. (1970), Multicommodity supply and transportation networks with resource constraints, The Rand Corporation, Report R. M.-6134-PR.

Глава 12

ПАРОСОЧЕТАНИЯ, ТРАНСПОРТНАЯ ЗАДАЧА И ЗАДАЧА О НАЗНАЧЕНИЯХ

1. Введение

Рассмотрим следующую задачу построения остовного подграфа G_p общего неориентированного графа G, степени вершин которого по отношению к подграфу G_p заданы.

Задача об остовном подграфе с предписанными степенями. Пусть $G = (X, A)$ — неориентированный граф, ребрам $a_j \in A$ которого приписаны веса c_j. Пусть, далее, заданы неотрицательные целые числа δ_i, $i = 1, \ldots, n$. Требуется найти в G такой остовный подграф G_p^*, что степени вершин x_i по отношению к G_p^* являются заданными числами δ_i (т. е. $d_i^{G_p^*} = \delta_i$) и сумма весов ребер графа G_p^* максимальна (или минимальна).

Сформулированную выше задачу мы будем называть в настоящей главе *общей задачей*. Очевидно, что если дан граф G и числа δ_i, то вполне возможно, что в G не существует никакого остовного подграфа G_p, имеющего эти предписанные степени. Два необходимых (но не достаточных, см. разд. 5) условия, которым должны удовлетворять числа δ_i, чтобы существовал некоторый допустимый остовный подграф, таковы:

$$\delta_i \leqslant d_i^G, \quad \forall i = 1, \ldots, n \tag{12.1}$$

■

$$\sum_{i=1}^{n} \delta_i \text{ — четна,}$$

причем последнее условие является прямым следствием того факта, что для любого неориентированного графа сумма степеней его вершин в два раза больше числа его ребер.

Паросочетанием общего неориентированного графа $G = (X, A)$ называется подмножество M множества A ребер графа G, выбранное так, что никакие два ребра из M не являются смежными (т. е. не имеют общей вершины). Следующая задача будет называться *задачей о максимальном паросочетании* (ЗМП).

ЗМП: найти паросочетание M^* максимального веса. Вес паросочетания определяется как

$$C_M = \sum_{a_j \in M} c_j.$$

M^* будет называться *максимальным паросочетанием* графа G. ЗМП можно выразить в терминах линейного программирования:

$$\text{максимизировать} \quad z = \sum_{j=1}^{m} c_j \xi_j \tag{12.2}$$

$$\text{при условии} \quad \sum_{j=1}^{m} b_{ij} \xi_j \leqslant 1 \quad \forall i = 1, \ldots, n, \tag{12.3}$$

$$\xi_j = 0 \quad \text{или} \quad 1, \tag{12.4}$$

где $[b_{ij}]$ — матрица инциденций графа G и $\xi_j = 1$ или 0 в зависимости от того, входит ли ребро a_j в паросочетание.

Совершенно очевидно, что ЗМП для графа $\hat{G}$ является частным случаем общей задачи. Если число вершин $\hat{G}$ четно, то требуется только добавлять к $\hat{G}$ ребра веса $-\infty$ до тех пор, пока не получится (новый) полный граф G. ЗМП становится тогда общей задачей для графа G, где все δ_i положены равными 1. Решением ЗМП будет решение общей задачи для (остовного) подграфа G_p^*, если пренебречь теми ребрами в G_p^*, которые имеют вес $-\infty$. Если число вершин графа $\hat{G}$ нечетно, то к $\hat{G}$ добавляют одну изолированную вершину, после чего строится граф G, как это было описано выше.

В терминологии гл. 3 паросочетание может быть названо «независимым множеством ребер», так как никакие два ребра паросочетания не являются смежными. «Доминирующее множество ребер» в смысле гл. 3 (т. е. множество ребер, «доминирующее» над всеми вершинами из G) следовало бы тогда определить как подмножество E ребер из G, выбранное так, что каждая вершина инцидентна по крайней мере одному ребру из E. Такое множество в литературе обычно называется *покрытием*, и в настоящей главе мы будем использовать этот термин. *Весом* покрытия E графа G называется $\sum_{a_j \in E} c_j$. Теперь ЗМП соответствует *задача о минимальном покрытии* (ЗПО). ЗПО: найти покрытие E^* минимального веса для графа G.

E^* будет называться *минимальным покрытием* графа G. Формулировка ЗПО в терминах линейного программирования такова:

$$\text{минимизировать} \quad z = \sum_{j=1}^{m} c_j \xi_j \tag{12.5}$$

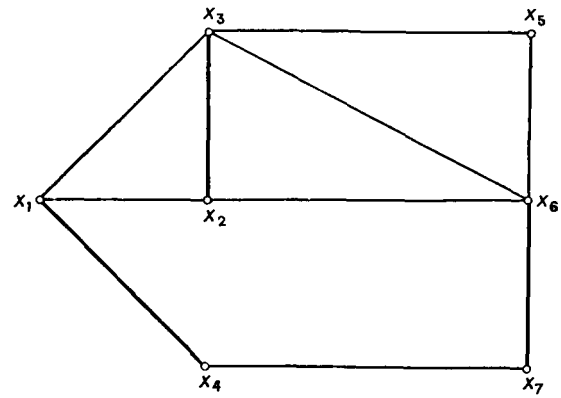
$$\text{при условии} \quad \sum_{j=1}^{m} b_{ij} \xi_j \geqslant 1, \quad \forall i = 1, \ldots, n, \tag{12.6}$$

$$\xi_j = 0 \quad \text{или} \quad 1, \tag{12.7}$$

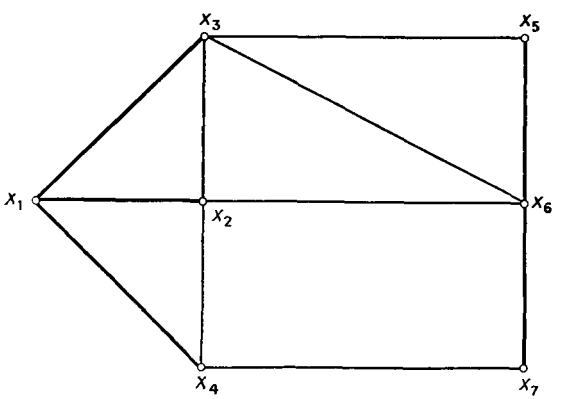
где $\xi_j = 1$ или 0 в зависимости от того, входит ли ребро a_j в покрытие.

На рис. 12.1 (а) паросочетание в графе показано жирными линиями, а на рис. 12.1 (б) в том же самом графе жирными линиями показано покрытие.

В частном случае, когда веса всех ребер равны единице, ЗМП и ЗПО сводятся к *задаче о наибольшем паросочетании* (ЗНПС) и



(а)



(б)

Рис. 12.1. (а) Паросочетание. (б) Покрытие.

задаче о покрытии наименьшей мощности (ЗПНМ). Если граф G имеет n вершин, то число ребер в паросочетании графа G не может, очевидно, превышать величины $[n/2]$. Однако это число не всегда достигается; например, для «звездного» графа на рис. 12.2 наибольшее число ребер в паросочетании равно 1.

В том специальном случае, когда веса c_j произвольны, а сам граф является двудольным, задача о максимальном паросочетании

сводится к *задаче о назначениях* (ЗН) — задаче, очень хорошо известной в исследовании операций. В графе такой специальной структуры общая задача превращается в другую хорошо известную задачу — *транспортную задачу* (ТЗ).

В настоящей главе мы дадим эффективные методы решения ЗМП ЗПО, ЗН и ТЗ. ЗМП возникает в качестве подзадачи в задаче об

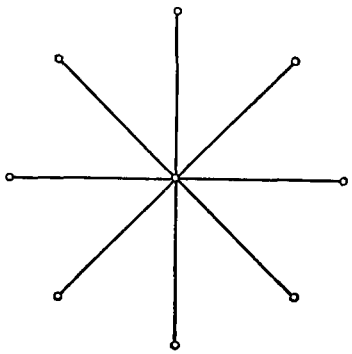


Рис. 12.2. Звездный граф.

обходе графа и в задаче китайского почтальона. Последние встречаются в задачах распределения, таких, как доставка молока и почты, посыпка зимой дорог смесью соли и песка, сбор мусора (см. гл. 9). ЗН и ТЗ являются двумя основными задачами с многочисленными прямыми и косвенными приложениями в самых различных областях, часто весьма далеких от тех областей, на которые указывают названия этих задач.

2. Наибольшие паросочетания

Если дано паросочетание M, то вершина x_i, не являющаяся конечной вершиной никакого ребра из M, будет называться *экспонированной* вершиной. На рис. 12.3, где паросочетание показано жирными линиями, вершины x_6 и x_9 являются экспозированными.

2.1. Альтернирующие цепи и деревья

Альтернирующая относительно M цепь — это простая цепь, ребра которой попеременно лежат или не лежат в паросочетании M. Примером такой цепи является цепь $(x_8, x_7, x_5, x_3, x_4, x_{10}, x_1)$ в графе на рис. 12.3. *Аугментальная относительно M цепь* — это такая альтернирующая цепь, начальная и конечная вершины

которой экспонированы. Цепь $(x_9, x_{10}, x_4, x_1, x_2, x_3, x_5, x_6)$ дает пример аугментальной цепи в графе, изображенном на рис. 12.3.

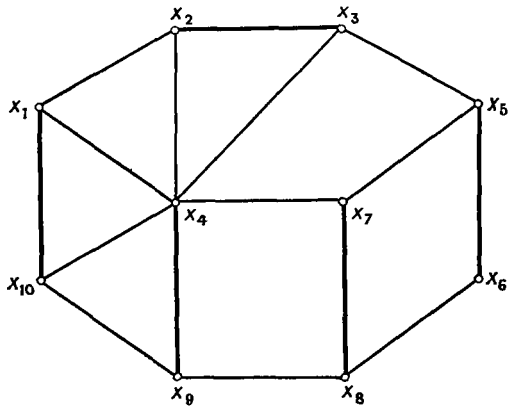


Рис. 12.3. Паросочетание M.

Основная относящаяся к паросочетаниям теорема может быть сформулирована так.

Теорема 1 [5, 27]. *Паросочетание M будет наибольшим паросочетанием тогда и только тогда, когда в G не существует никакой аугментальной относительно M цепи.*

Доказательство. Необходимость. Допустим, что существует аугментальная цепь P, и пусть P обозначает также множество ребер этой цепи. Никакое ребро в множестве $M - P \cap M$ не может быть смежным ни с каким ребром в P, так как две концевые вершины в P являются (по определению) экспонированными, а каждая другая вершина в P уже является концевой вершиной некоторого ребра в $P \cap M$. Следовательно, множество ребер $M' = (M - P \cap M) \cup (P - P \cap M) = M \cup P - P \cap M$, полученное в результате замены тех ребер из P, которые не входят в M (т. е. ребер из множества $P - P \cap M$), на ребра из P, которые входят в M (т. е. на ребра из $P \cap M$), является паросочетанием. Но так как оба концевых ребра из P не лежат в M, то $P - P \cap M$ содержит на одно ребро больше, чем $P \cap M$, и поэтому $|M'| = |M| + 1$, так что M не является наибольшим паросочетанием.

Достаточность. Пусть M — паросочетание, относительно которого нет ни одной аугментальной цепи в графе G, и пусть M^* — наибольшее паросочетание. Построим остовный подграф G_p, образованный ребрами из $M \cup M^* - M \cap M^*$ вместе с соответствующими им вершинами. Никакая вершина из G_p не может иметь

степень, большую чем 2, так как в противном случае два или более ребер из M (или M^*) являются смежными, что противоречит определению паросочетания. Таким образом, подграф G_p состоит из одной или более компонент, и каждая из них есть либо изолированная вершина, либо простая цепь, либо простой цикл, как это изображено на рис. 12.4.

Цепь типа (б) существовать не может, так как она является аугментальной цепью относительно M, это противоречит перво-

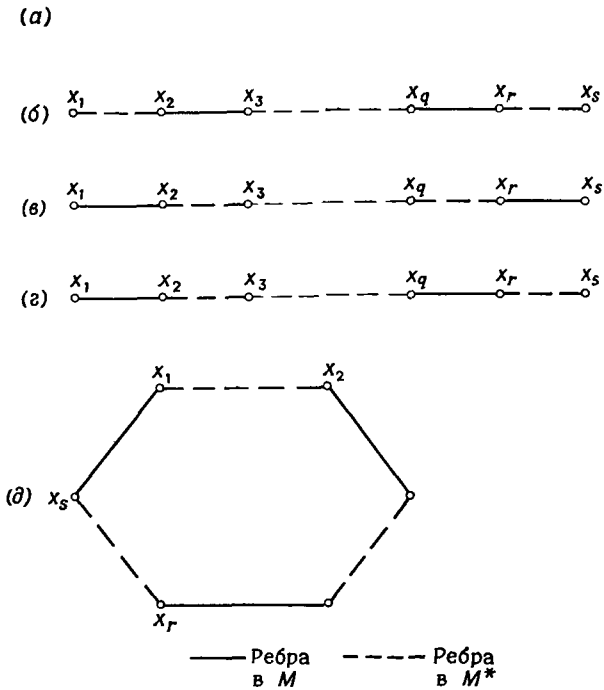


Рис. 12.4.

начальному предположению. Цепь типа (в) не может существовать, так как она является аугментальной относительно M^*, что противоречит предположению о максимальности M^*. Цикл типа (д) с нечетным числом ребер не может существовать, так как тогда либо два ребра из M, либо два ребра из M^* были бы смежными друг с другом. Остаются возможными графы следующих видов: (а), (г) и циклы типа (д) с четным числом ребер. Для каждого из таких графов число ребер $n(M)$ в M равно числу ребер $n(M^*)$ в M^*. Так как это справедливо для любой связной компоненты k

графа G_p, то

$$\sum_k n_k(M) = \sum_k n_k(M^*),$$

где $n_k(M)$ и $n_k(M^*)$ — число ребер в компоненте k, принадлежащих соответственно M и M^*. Поэтому

$$|M| \equiv |M \cap M^*| + \sum_k n_k(M) = |M \cap M^*| + \sum_k n_k(M^*) \equiv |M^*|$$

и, следовательно, M — наибольшее паросочетание.

Рассмотрим в качестве примера граф, изображенный на рис. 12.3. Ребра, образующие паросочетание M в этом графе, выделены жирными линиями. Множество ребер в аугментальной

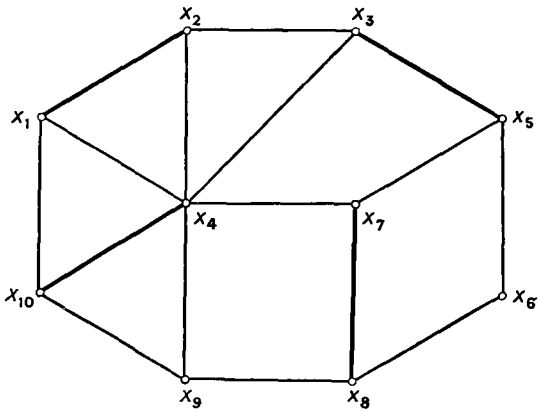


Рис. 12.5. Наибольшее паросочетание M'.

цепи $P = (x_9, x_4, x_{10}, x_1, x_2, x_3, x_5, x_6)$, не лежащих в M, дает новое паросочетание M', изображенное на рис. 12.5. Здесь уже нет экспонированных вершин и поэтому не существует никаких аугментальных относительно M' цепей. Следовательно, по теореме 1 M' — наибольшее паросочетание.

Альтернирующим деревом относительно данного паросочетания M называется дерево T, для которого

(а) одна вершина, называемая *корнем* дерева T, является экспонированной;

(б) все начинающиеся в корне цепи являются альтернирующими;

(в) все максимальные цепи, начинающиеся в корне дерева T, содержат четное число ребер.

Разобъем все вершины дерева на два класса: Ф — класс *внешних* вершин, I — класс *внутренних* вершин. Корень дерева отне-

сен к классу Ф. Вершины вдоль любой цепи, начинающейся в корне, будут поочередно отнесены к классам Ф и I. Таким образом, если вершина расположена в «конце» цепи с нечетным числом ребер, начинающейся в корне, то эта вершина будет отнесена к I, а если она лежит в «конце» цепи с четным числом ребер, начинающейся в корне, то будет отнесена к Ф. На рис. 12.6 изображено альтернирующее дерево. Здесь все вершины, лежащие в конце максимальных цепей, начинающихся с корня, в соответствии

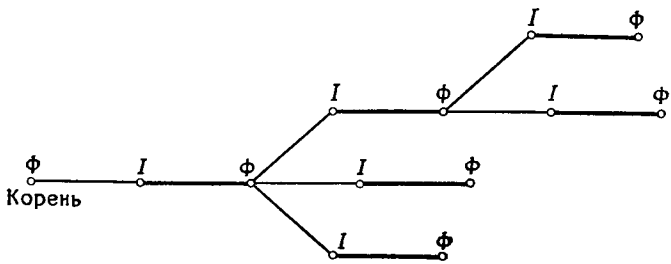


Рис. 12.6. Альтернирующее дерево. I — внутренние вершины, Ф — внешние вершины.

с условием (в) будут «внешними». Степень любой внутренней вершины альтернирующего дерева равна 2, в то время как степень внешней вершины может быть любым целым положительным числом.

Аугментальное дерево — это альтернирующее дерево относительно данного паросочетания, удовлетворяющее условию: существует ребро от какой-нибудь внешней вершины дерева x_o, до некоторой экспонированной вершины x_e, не принадлежащей дереву. Единственная цепь, идущая от корня дерева до вершины x_o и далее — по ребру (x_o, e_e), будет тогда аугментальной цепью.

2.2. Цветки

Цветком по отношению к паросочетанию M называется аугментальная цепь, начальная и конечная экспонированные вершины которой совпадают, т. е. эта цепь является циклом, так как число ребер этой цепи нечетно. На рис. 12.7 изображен цветок, образованный цепью с 5 ребрами.

В алгоритмах для ЗМП и ЗНПС, описанных ниже, цветки *срезают* для того, чтобы получить более простой новый граф. *Срезание* цветка B состоит в замене всех вершин цветка B (скажем, X_B) одной новой *псевдовершиной* x_b. Ребро (x_b, x_k) добавляется всегда, когда существует ребро из некоторой вершины $x_i \in X_B$ к другой вершине $x_k \notin X_B$. В упрощенном графе, полу-

ченном после срезания, вершина x_b и другие псевдовершины, соответствующие ранее срезанным цветкам, могут в свою очередь образовывать новый цветок, который опять срезается, и т. д. Последний цветок B_0, не содержащийся ни в каких других цветках, называется *крайним цветком*. Обоснование процесса срезания цветков основано на теореме 2, приводимой ниже.

Цветущее дерево — это альтернирующее дерево относительно данного паросочетания, для которого существует ребро (x_0', x_0''), соединяющее две внешние вершины дерева: x_0' и x_0''. Если P' — множество ребер цепи, начинающейся в корне альтернирующего

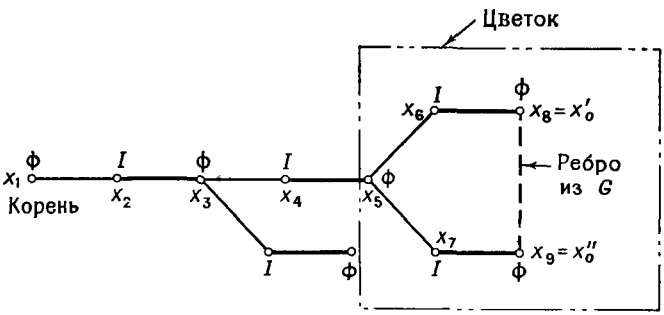


Рис. 12.7. Дерево с цветком.

дерева и кончающейся в x_0', а P'' — соответствующее множество для x_0'', то множество ребер $[P' \cup P'' - P' \cap P'']$ вместе с ребром (x_0', x_0'') образует цветок. На рис. 12.7 изображено цветущее дерево.

Когда цветок B срезан, получающаяся псевдовершина считается внешней вершиной. Всякая вершина из B может быть по желанию отнесена к внутренним или к внешним вершинам, так как если вершина принадлежит B, то она является концевой и в цепях с четным, и в цепях с нечетным числом ребер, начинающихся в корне дерева (в зависимости от маршрута, по которому проходят эти цепи до прихода в первую вершину цветка). Так, на рис. 12.7 вершина x_6 может быть помечена как внутренняя, если рассматривать цепь $(x_1, x_2, x_3, x_4, x_5, x_6)$, и как внешняя, если брать цепь $(x_1, x_2, x_3, x_4, x_5, x_7, x_9, x_8, x_6)$. Но когда псевдовершина, получающаяся после срезания цветка, помечается как «внешняя», структура остающегося альтернирующего дерева все еще будет корректной — как это вытекает из определения альтернирующего дерева. Таким образом, после срезания цветка альтернирующее дерево в получившемся графе сохраняется.

Пример сложной формации цветков и их срезание показаны на рис. 12.8а — 12.8г. В графе на рис. 12.8а паросочетание

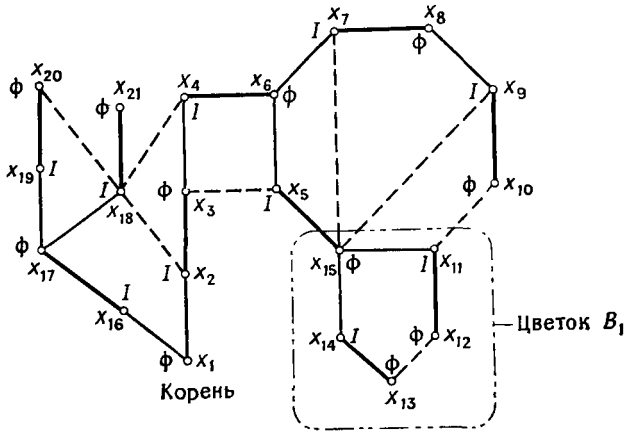


Рис. 12.8а. Срезание цветка B_1.
— Ребра в альтернирующем дереве, входящие в паросочетание.
— Ребра в альтернирующем дереве, не входящие в паросочетание
— — — Другие ребра графа.

изображено жирными линиями. Альтернирующее дерево T с корнем в экспонированной вершине x_1 показано сплошными линиями, а все другие ребра графа, не входящие в дерево T,— пунктиром. Допустим, что последнее добавленное к дереву ребро есть (x_{11}, x_{12}). Так как между двумя внешними вершинами x_{12} и x_{13} существует ребро, то добавление к дереву вышеупомянутого последнего ребра дает цветок. Цветок B_1, обведенный на рис. 12.8а штрих-

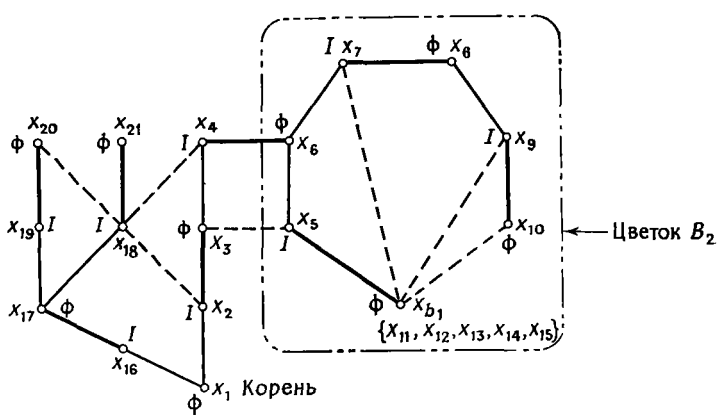


Рис. 12.8б. Срезание B_2.

пунктирной замкнутой линией, после срезания дает псевдовершину x_{b_1}, как это показано на рис. 12.8б. Дерево на рис. 12.8б все еще является цветущим с цветком B_2, так как существует ребро (x_{10}, x_{b_1}). После срезания цветка B_2 «остается» вершина x_{b_2} и получается новый граф, изображенный на рис. 12,8в. Дерево на рис. 12.8в опять будет цветущим ввиду существования ребра (x_3, x_{b_2}); цветком в нем будет B_3. После срезания цветка

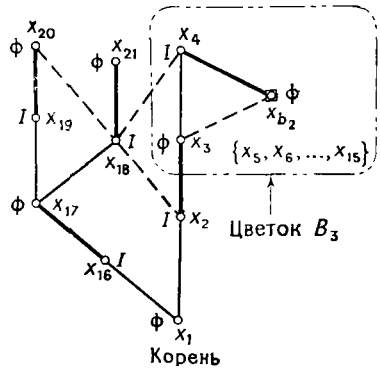


Рис. 12.8в. Срезание B_3.

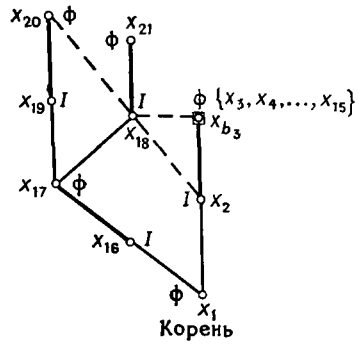


Рис. 12.8г. Альтернирующее дерево после срезания цветков.

B_3 «остается» вершина x_{b_3} и граф, изображенный на рис. 12.8г. Больше цветков нет, и, значит, B_3 будет крайним цветком.

Теорема 2. *Если B — цветок с нечетным множеством вершин X_B и если x — некоторая вершина из X_B, то в подграфе $\langle X_B \rangle$ существует наибольшее паросочетание, для которого вершина x будет экспонированной.*

Доказательство. Пусть $(x_1, x_2, \ldots, x_\alpha)$ — цикл с нечетным числом ребер, образующий цветок B. Тогда или $x = x_i$ для некоторого $i = 1, 2, \ldots, \alpha$, или же $x \in x_i$ для некоторой псевдовершины x_i, соответствующей ранее срезанному цветку B_i.

В первом случае вершина x может быть оставлена экспонированной, а остальные $\alpha - 1$ вершин из B разбиваются на пары, образуя паросочетание. Такое разбиение возможно, так как $\alpha - 1$ — четное число вершин, остающихся в цикле (после выбора вершины x). Если одна из этих вершин, скажем x_p, соответствует ранее срезанному цветку B_p, то вхождение x_p в паросочетание будет означать возможность разбиения на пары всех вершин из B_p, кроме одной, а одна уже «использована» на предыдущем шаге. Этот процесс можно продолжить до тех пор, пока все вершины (отличные от x) не будут разбиты на пары, образуя паросочетание.

В последнем из двух упомянутых случаев вершина x_i может быть оставлена экспонированной, а остальные $\alpha - 1$ вершин из B разобьются на пары, порождая паросочетание, как и выше. Цветок B_i — это цикл с нечетным числом ребер, и, следовательно, получается такая же самая ситуация относительно B_i, какая уже обсуждалась в случае цветка B, так что экспонированной в конце концов останется только вершина x.

Важность цветков и циклов (цепей) с нечетным числом ребер в задачах о паросочетаниях можно оценить, анализируя следующий факт: при отсутствии таких циклов формулировка ЗМП на языке линейного программирования, даваемая соотношениями (12.2) и (12.3), приводит к собственно целочисленному решению, так что ограничения (12.4) становятся излишними [8, 16, 17]. Графы. не содержащие циклов с нечетным числом ребер, являются двудольными графами; они рассматриваются в последующих параграфах. Общие свойства полиэдров, определяемых соотношениями (12.3), содержатся в следующей теореме Балински [1].

Теорема 3. *Все вершины выпуклого полиздра, задаваемого неравенствами*

$$\sum_{j=1}^{m} b_{ij}\xi_j \leqslant 1, \qquad \forall i = 1, \ldots, n,$$

где $[b_{ij}]$ — *матрица инциденций графа, имеют координаты со значениями* 0, 1/2 *или* 1.

Случай нецелочисленных значений (а именно значения 1/2) отвечает циклам с нечетным числом ребер. Если, например, граф

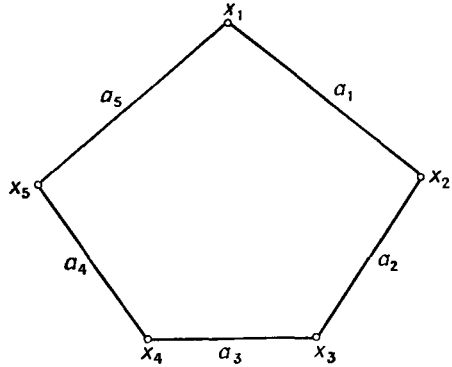


Рис. 12.9.

состоит из единственного такого цикла, как показано на рис. 12.9, то решение задачи линейного программирования (12.2) и (12.3)

для ЗНП таково: $\xi_j = 1/2$ для $j = 1, \ldots, 5$. Значение этого решения будет $2\frac{1}{2}$, в то время как очевидно, что наибольшее паросочетание для графа на рис. 12.9 содержит два ребра.

2.3. Венгерские деревья

Венгерское дерево — это такое альтернирующее дерево в графе, что каждое ребро графа, имеющее внешнюю вершину дерева в качестве концевой, имеет другой концевой вершиной внутреннюю вершину этого дерева. На рис. 12.10 сплошными линиями

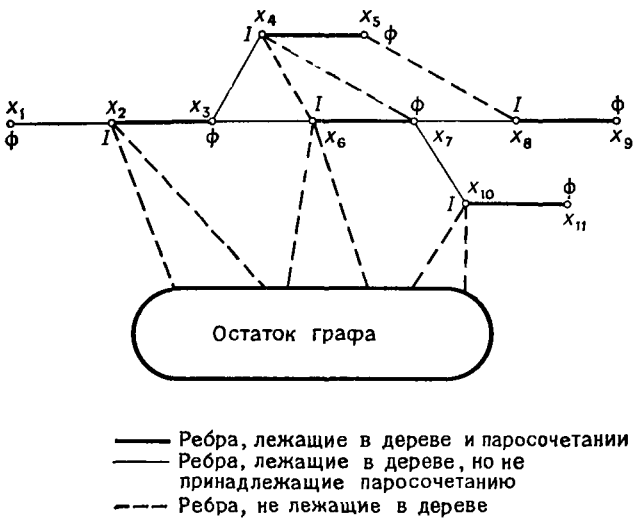


Рис. 12.10. Венгерское дерево.

показано венгерское дерево, причем жирными линиями изображены ребра, входящие в паросочетание, а светлыми — остальные. Ребра графа, не принадлежащие дереву, показаны пунктиром.

Важность венгерских деревьев для ЗМП объясняется следующим их свойством.

Теорема 4 [10]. *Пусть H — венгерское дерево в графе $G = (X, A)$ и $G_o = \langle X - X_H \rangle$ — порожденный подграф графа G, «исключающий» из G множество вершин X_H дерева H. Если M_H — паросочетание в дереве H и $M^*_{G_o}$ — наибольшее паросочетание в G_o, то множество ребер $M_H \cup M^*_{G_o}$ является наибольшим паросочетанием в G.*

Доказательство. Пусть множество A ребер графа G разбито на три подмножества:

$$A_H = \{(x_i, x_j) \mid (x_i, x_j) \in A \quad \text{и} \quad x_i, x_j \in X_H\},$$
$$A_{HG_0} = \{(x_i, x_j) \mid (x_i, x_j) \in A, x_i \in X_H \quad \text{и} \quad x_j \in X - X_H\},$$
$$A_{G_0} = \{(x_i, x_j) \mid (x_i, x_j) \in A \quad \text{и} \quad x_i, x_j \in X - X_H\}.$$

Если S — произвольное паросочетание в G, то S может быть разбито аналогичным образом, так что

$$S = S_H \cup S_{HG_0} \cup S_{G_0},$$

где $S_H = S \cap A_H$, $S_{HG_e} = S \cap A_{HG_0}$ и $S_{G_0} = S \cap A_{G_0}$.

По определению паросочетания $M^*_{G_0}$ мы имеем

$$|M^*_{G_0}| \geqslant |S_{G_0}|. \tag{12.8}$$

Рассмотрим теперь граф G', состоящий из ребер $A_H \cup A_{HG_0}$ и их концевых вершин. По отношению к M_H все вершины $x_k \in X - X_H$, являющиеся концевыми для ребер из A_{HG_0}, будут экспонированными. Альтернирующая цепь, начинающаяся в экспонированной вершине $x_k \in X - X_H$ (или начинающаяся в корне дерева H, который также является экспонированной вершиной), может только тогда быть аугментальной, когда она оканчивается в другой вершине $x_k \in X - X_H$. Но аугментальная цепь должна содержать нечетное число ребер, а это означает, что если первое ребро этой цепи идет из экспонированной вершины во внутреннюю, то последнее ребро должно идти из внешней вершины в экспонированную. Так как по определению дерева H все ребра в A_{HG_0} лежат между внутренними вершинами H и вершинами из $X - X_H$, то в графе G' нет никакой аугментальной цепи и, следовательно, M_H является наибольшим паросочетанием в G', т. е.

$$|M_H| \geqslant |S_H| + |S_{HG_0}|. \tag{12.9}$$

Из (12.8) и (12.9) получаем

$$|M_H \cup M^*_{G_0}| \geqslant |S_H \cup S_{HG_0} \cup S_{G_0}| \geqslant |S|. \tag{12.10}$$

Таким образом, $M_H \cup M^*_{G_0}$ является наибольшим паросочетанием в G.

2.4. Алгоритм для ЗНПС

Пусть задано начальное паросочетание. (Допустимо использование пустого паросочетания.) Алгоритм осуществляет систематический поиск аугментальных цепей с целью улучшения заданного паросочетания в соответствии с теоремой 1. Алгоритм был предложен Эдмондсом [10, 11].

Альтернирующее дерево имеет своим корнем некоторую экспонированную вершину и строится путем попеременного добавления

ребер, лежащих и не лежащих в паросочетании, до тех пор, пока или

(I) дерево станет аугментальным, или

(II) на дереве появится цветок, или

(III) дерево станет венгерским.

В случае (I) число ребер в паросочетании может быть увеличено на единицу с помощью движения по аугментальной цепи к корню дерева и замены ребер цепи, принадлежащих паросочетанию, на ребра, не принадлежащие ему, и наоборот [1]). После этого дерево отбрасывается [2]) и строится новое дерево, если такое существует, с корнем в некоторой оставшейся экспозированной вершине.

В случае (II) обнаруживается получившийся цветок (как описано в разд. 2.2), срезается и продолжается рост дерева в процессе поиска аугментальной цепи. Что касается вычислительной стороны, то срезание не производится «явным образом». Достаточно пометить все вершины цветка как внешние и отметить, что все они принадлежат этому цветку. Порядок, в котором цветки «срезаются», важен, так как в конце всей процедуры цветки должны «расцвести» в обратном порядке.

В случае (III) вершины венгерского дерева и инцидентные им ребра совсем удаляются из графа (в соответствии с теоремой 4) и алгоритм применяется к оставшемуся подграфу.

2.4.1. Описание алгоритма

Начальная установка.

Шаг 1. Выбрать в G начальное паросочетание M.

Выбор корня.

Шаг 2. Если в G существуют по крайней мере две экспонированные вершины, то взять одну из них в качестве корня; в противном случае перейти к шагу 8.

Шаг 3. Взять внешнюю вершину x_o дерева. Для каждого ребра (x_o, x_i): если x_i — экспонированная вершина, то перейти к шагу 4; если x_i — внутренняя вершина, то перейти к шагу 7; если x_i не принадлежит дереву и не является экспонированной, то перейти к шагу 5.

---

[1]) При описанном построении получается новое паросочетание, состоящее из тех ребер выбранной аугментальной цепи, которые не принадлежали первоначальному паросочетанию.— *Прим. ред.*

[2]) Дерево строится «отдельно от графа» и *из графа не выбрасывается*; если при построении дерева оно просто как-то выделялось (помечалось) в графе, то «выбрасывание» состоит в устранении всяких пометок, приписанных элементам дерева в этом процессе.— *Прим. ред.*

Шаг 4. Найдена аугментальная цепь из корня к x_i. Построить новое улучшенное паросочетание, удаляя текущее дерево и пометки вершин, и перейти к шагу 2.

Шаг 5. Добавить к альтернирующему дереву ребро (x_o, x_i) и отметить вершину x_i как внутреннюю. Найти ребро (x_i, x_k), принадлежащее текущему паросочетанию, и добавить его к дереву, пометив вершину x_k как внешнюю. Если существует ребро между x_k и другой внешней вершиной, то перейти к шагу 6. В противном случае перейти к шагу 3.

Шаг 6. Опознать и срезать получившийся цветок. Отметить возникшую псевдовершину как внешнюю. Внести поправку в частичное упорядочение цветков и возвратиться к шагу 3.

Шаг 7. Возвращаться к шагу 3 до тех пор, пока единственным возникающим случаем не будет случай (III). Тогда удалить все вершины дерева и все ребра из G, инцидентные этим вершинам. Считать оставшийся подграф графом G и вернуться к шагу 2.

Шаг 8. Выделить отдельно в последнем графе G и в каждом удаленном венгерском графе оптимальное паросочетание, поступая так. Распустить сначала крайний цветок и выделить паросочетание, которое оставляет экспонированной относительно него ту вершину x, которая входит в паросочетание в нераспустившемся цветке. (См. теорему 2.) Продолжать процесс распускания в порядке, обратном к установленному на шаге 6, до тех пор, пока весь граф не будет распустившимся и не будет получено наибольшее паросочетание.

2.5. Пример

Мы хотим найти наибольшее паросочетание в графе, изображенном на рис. 12.11а, начальное паросочетание в котором показано жирными линиями. Мы применим алгоритм предыдущего раздела с тем правилом, что на шаге 3 внешние вершины просматриваются в порядке увеличения их индексов и если дана внешняя вершина, то ребра, ведущие из нее, просматриваются в порядке увеличения индексов их вторых концевых вершин.

На шаге 2 алгоритма в качестве корня выбирается вершина x_1, и после нескольких применений шагов 3 и 5 получается альтернирующее дерево, изображенное на рис. 12.11б. На этой стадии сразу же после добавления ребер (x_3, x_8) и (x_8, x_9) к альтернирующему дереву шаг 6 опознает цветок $(x_3, x_8, x_9, x_5, x_4)$, дающий после срезания псевдовершину b_1, и приводит к дереву, изображенному на рис. 12.11в.

Следующее применение шага 3 добавляет ребра (b_1, x_{10}) (на самом деле (x_4, x_{10})) и (x_{10}, x_{11}), как показано на рис. 12.11 г. Второй цветок $(b_1, x_{10}, x_{11}, x_7, x_6)$ опознается на шаге 6. После

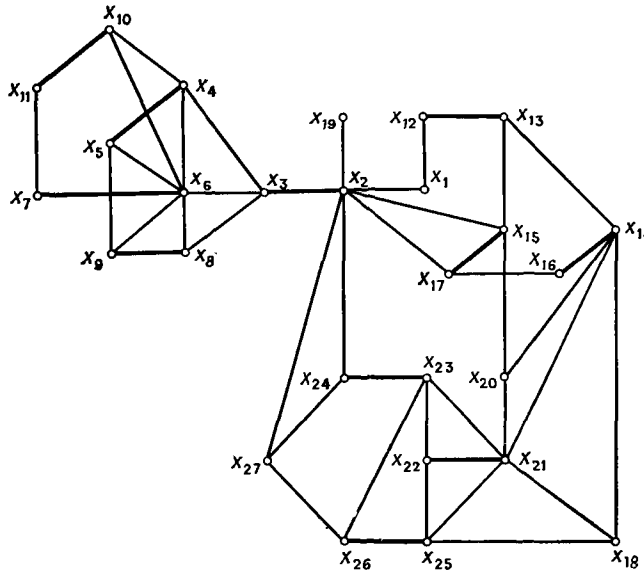


Рис. 12.11а. Граф из примера 2.5. — Ребра в паросочетании.

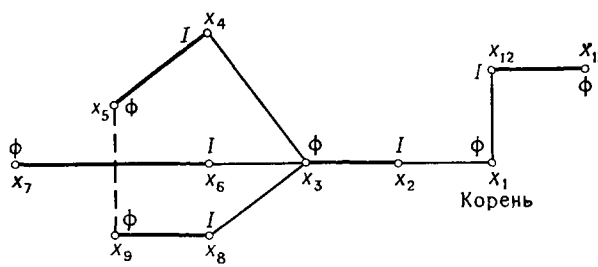


Рис. 12.11б. Альтернирующее дерево в графе из рис. 12.11а.

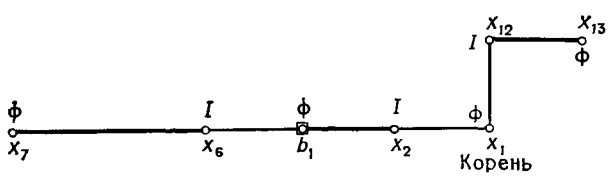


Рис. 12.11в. Дерево после срезания цветка b_1.

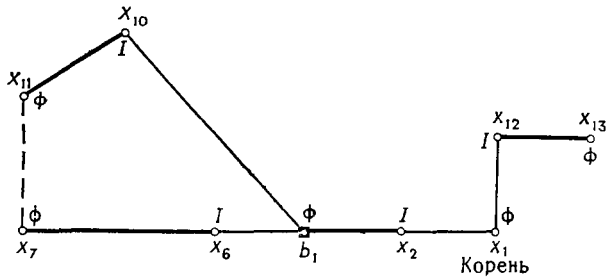


Рис. 12.11г. Образовался цветок b_2.

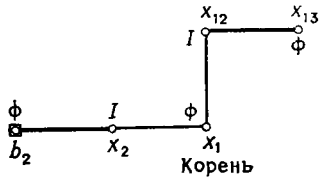


Рис. 12.11д. Дерево после срезания цветка b_2.

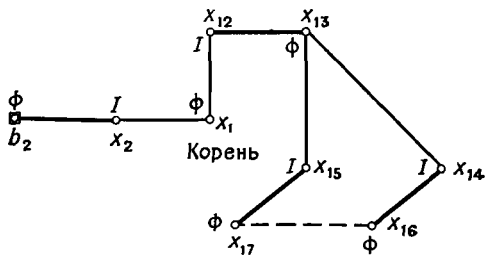


Рис. 12.11е. Образование цветка b_3.

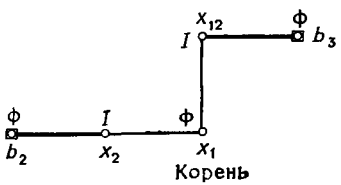


Рис. 12.11ж. Дерево после срезания b_3.

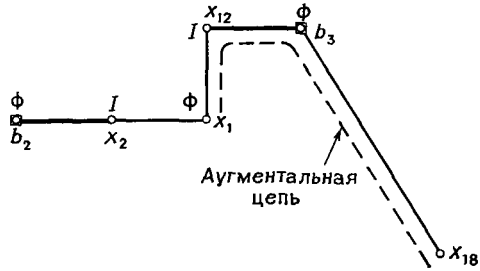


Рис. 12.11з. Найдена аугментальная цепь.

его срезания возникает псевдовершина b_2 и образуется дерево, показанное на рис. 12.11д. Рост дерева продолжается из вершины x_{13}, но после добавления ребер (x_{13}, x_{14}), (x_{14}, x_{16}), (x_{13}, x_{15}), (x_{15}, x_{17}) опознается третий цветок $(x_{13}, x_{14}, x_{16}, x_{17}, x_{15})$, показанный на рис. 12.11е. После его срезания возникает псевдовершина b_3 и образуется дерево, изображенное на рис. 12.11ж.

При следующем применении шага 3 добавляется ребро (b_3, x_{18}) и на шаге 4 обнаруживается аугментальная цепь $(x_1, x_{12}, b_3, x_{18})$ (см. рис. 12.11з). Когда распускается b_3 (см. рис. 12.11и), то возможны две цепи от x_1 к x_{18}. Аугментальной цепью будет та цепь, которая состоит из нечетного числа ребер, т. е. $P_2 = (x_1, x_{12}, x_{13}, x_{15}, x_{17}, x_{16}, x_{14}, x_{18})$. Когда ребра этой цепи, лежащие в паросочетании, «меняются» с теми, которые в нем не лежат, то получается новое улучшенное паросочетание, изображенное на рис. 12.12а.

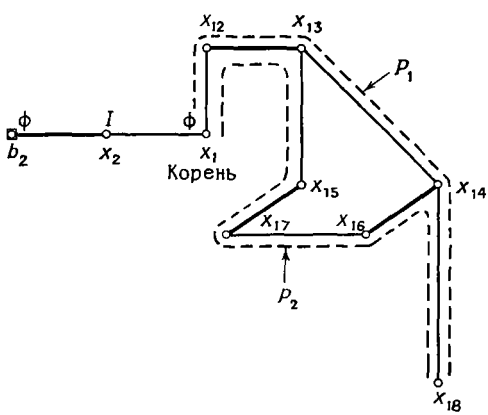


Рис. 12.11и. Цветок b_3 распускается. P_2 — аугментальная цепь.

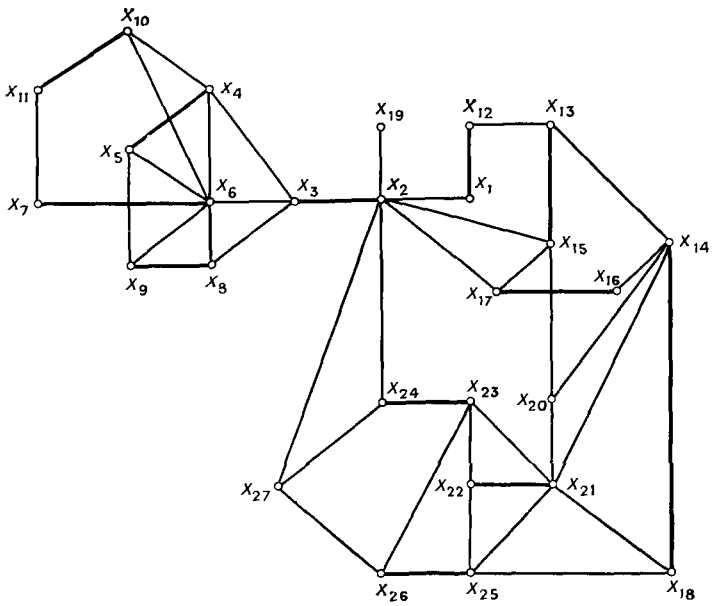


Рис. 12.12а. Улучшенное паросочетание.

«Отбросив» дерево [1]) и начиная с нового паросочетания, выберем на шаге 2 в качестве корня вершину x_{19}. После нескольких применений шагов 3, 5 и 6 получается дерево, показанное на рис. 12.12б, где b_2 имеет тот же самый смысл, что и ранее. На шаге 7 обнаруживается, что это дерево является венгерским. После его удаления остается подграф, изображенный на рис. 12.13а.

На шаге 2 в качестве нового корня выбирается x_{20}. После нескольких применений шагов 3 и 5 — сразу же после добавления ребер (x_{18}, x_{25}) и (x_{25}, x_{26}) на шаге 5 — обнаруживается аугментальная цепь. Окончательный вид дерева показан на рис. 12.13 б; аугментальной цепью является цепь $(x_{20}, x_{14}, x_{18}, x_{25}, x_{26}, x_{27})$. «Поменяв» ребра этой цепи, лежащие в паросочетании на те, которые

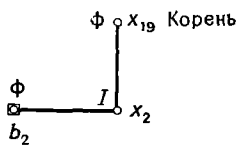


Рис. 12.12б. Венгерское дерево в графе из (а).

[1]) См. примечание 2 на стр. 382.— *Прим. ред.*

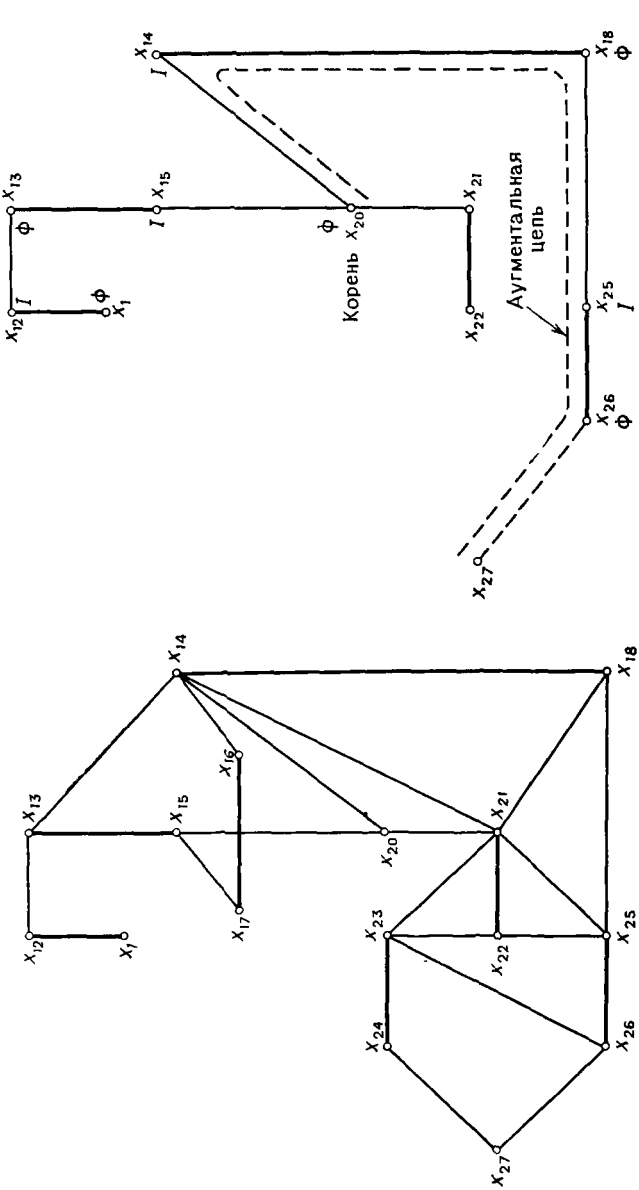


Рис. 12.13. (а) Граф после удаления венгерского дерева. (б) Альтернирующее дерево и аугментальная цепь в графе из (а).

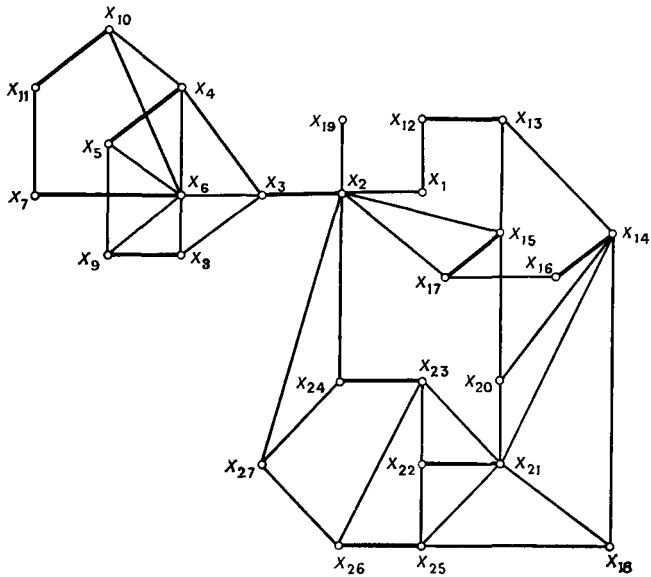


Рис. 12.14. Наибольшее паросочетание в примере 2.5.

в нем не лежат, получим паросочетание, не оставляющее никаких экспозированных вершин. (Заметим, что вершина x_{19} удалена ранее вместе с венгерским деревом.) Получающееся паросочетание, изображенное на рис. 12.14, будет, таким образом, наибольшим паросочетанием.

3. Максимальные паросочетания

Рассмотрим теперь задачу о паросочетании для общего графа $G = (X, A)$, ребрам которого $a_j \in A$ приписаны веса c_j. В этом разделе мы исследуем задачу нахождения *совершенного* паросочетания (т. е. паросочетания, в котором каждая вершина «сочетается» с некоторой другой вершиной), имеющего максимальный вес. Эта задача определяется соотношениями (12.2) — (12.4) с той лишь разницей, что неравенство в (12.3) должно быть заменено равенством. Во введении было объяснено, как задачи с ограничениями в виде равенств или неравенств можно сделать эквивалентными путем добавления ребер с весом $-\infty$ и искусственной вершины — в случае нечетного числа вершин. Таким образом, рассматриваемая ЗМП формулируется так:

максимизировать

$$z = \sum_{j=1}^{m} c_j \xi_j \tag{12.11}$$

при условии

$$\sum_{a_j \in T_i} \xi_j = 1, \qquad \forall i = 1, \ldots, n, \tag{12.12}$$

где

$$T_i = \{(x_i, x_k) \mid x_k \in \Gamma(x_i)\},$$

т. е. просто множество ребер, инцидентных вершине x_i, и

$$\xi_j \in \{0, 1\}. \tag{12.13}$$

Как уже говорилось выше, условия целочисленности (12.13) не являются излишними в случае существования в графе циклов с нечетным числом ребер. Однако Эдмондс [10, 11, 12] показал, что эти условия могут быть заменены системой линейных ограничений, и доказал следующую теорему.

Теорема 5 [11]. *Для любого графа выпуклая оболочка решений* (12.12) *и* (12.13) *является полиэдром, определенным соотношениями* (12.12), *и, кроме того*:

(i) *для каждого подмножества* $S_r \subset X$, *содержащего нечетное число вершин* (*скажем,* $|S_r| = 2q_r + 1$ *с некоторым положительным целым* q_r), *следует добавить ограничения*

$$\sum_{a_j \in A_r} \xi_j \leqslant q_r, \tag{12.14}$$

где A_r — *множество ребер порожденного подграфа* $\langle S_r \rangle$, *и*

(ii) *следует добавить ограничения* $\xi_j \geqslant 0$ *для* (12.15) *всех* $j = 1, \ldots, m$.

Ясно, что любое паросочетание из G удовлетворяет ограничениям (12.14). Не очевидно только, что эти ограничения достаточны. Мы обоснуем их достаточность конструктивно, предъявив совершенное паросочетание из G, являющееся решением задачи линейного программирования (12.11), (12.12), (12.14) и (12.15) с любым заданным вектором $[c_j]$ реберных весов.

Обозначим сформулированную выше задачу линейного программирования через P. Двойственной задачей к P будет [8] следующая:

минимизировать

$$u = \sum_{i=1}^{n} \pi_i + \sum_{S_r} q_r \lambda_r \tag{12.16}$$

при условии

$$\pi_i + \pi_k + \sum_{S_r \in F_j} \lambda_r \geqslant c_j, \qquad \forall j = 1, \ldots, m \tag{12.17}$$

и

$$\lambda_r \geqslant 0 \quad \text{для всех } S_r. \tag{12.18}$$

Таким образом, с каждой вершиной x_i из G связана некоторая переменная π_i, а с каждым множеством S_r, состоящим из нечетного числа вершин, связана переменная λ_r.

В ограничениях (12.17) c_j — вес ребра a_j вида (x_i, x_k), а $F_j = \{S_r \mid a_j \in A_r\}$, т. е. F_j — семейство подмножеств S_r, содержащих как x_i, так и x_k. В соответствии с теоремой двойственности линейного программирования вектор $[\xi_j^*]$ максимизирует z при условиях (12.12), (12.14) и (12.15), а вектор $[\pi_i^*, \lambda_r^*]$ минимизирует u при условиях (12.17) и (12.18) тогда и только тогда, когда

(i) для каждого λ_r^* или $\lambda_r^* = 0$, или же в соответствующем ограничении (12.14) имеет место равенство;

(ii) для каждого ξ_j^* или $\xi_j^* = 0$, или же в соответствующем ограничении (12.17) имеет место равенство.

Мы дадим доказательство теоремы 5, описав процедуру, позволяющую находить совершенное паросочетание M (и, следовательно, вектор $[\xi_j]$, а также вектор $[\pi_i^*, \lambda_r^*]$, удовлетворяющий условиям (12.17) и (12.18) и условиям двойственности (i) и (ii)). Таким образом будет доказано, что M — оптимальное паросочетание.

3.1. Алгоритм для ЗМП

Предположим, что мы начинаем с графа G. Присвоим его вершинам веса π_i, выбирая величины π_i достаточно большими, чтобы ограничения (12.17) выполнялись, когда все λ_r взяты равными нулю. Пусть теперь G' — остовный подграф в G, содержащий только те ребра, для которых в (12.17) имеет место равенство. Назовем G' *специальным остовным подграфом* графа G. Если в G' можно найти совершенное паросочетание (с помощью алгоритма для ЗНПС, описанного в предыдущем разделе), то это паросочетание — оптимальное. Действительно, вектор $[\pi_i, \lambda_r]$ удовлетворяет ограничениям (12.17) и (12.18), условие (i) выполнено автоматически, так как $\lambda_r = 0$ для всех r, а условие (ii) выполнено ввиду того, что в паросочетание входят только ребра из G', так что ξ_j не может быть $\neq 0$, если a_j не удовлетворяет ограничению (12.17).

Вообще говоря, может оказаться, что в G' совершенное паросочетание найти нельзя, а алгоритм для ЗНПС обнаружит венгерское дерево. Как уже отмечалось в разд. 2.3, наибольшее паросоче-

тание в G' частично строится из паросочетания в H, оставляющего экспонированным корень дерева H, и поэтому существование венгерского дерева означает, что в G' нет никакого совершенного паросочетания. Таким образом, при обнаружении такого дерева следует изменить вектор весов $[\pi_i, \lambda_r]$, чтобы получить новый специальный остовный подграф G' в графе G и подвергнуть его проверке. Ниже будет описано, как вычисляется такой весовой вектор.

Допустим на время, что для любого допустимого вектора весов $[\pi_i, \lambda_r]$ (т. е. вектора, удовлетворяющего (12.17) и (12.18)) данное λ_r является ненулевым лишь тогда, когда S_r будет множеством вершин, содержащихся в псевдовершине текущего графа G' или в псевдовершине, содержащейся в псевдовершине графа G' на любом уровне процесса срезания. Допустим, кроме того, что для всех ребер, образующих на некотором этапе цветки, позднее срезанные и давшие псевдовершины графа G', в выражении (12.17) имеет место равенство. Два сделанных выше допущения наверняка выполняются вначале, так как первый специальный остовный подграф G', построенный, как описано выше, вообще не имеет псевдовершин и для него все λ_r положены равными нулю.

Для более ясного понимания алгоритма мы введем следующую индексацию. Первоначальный граф G будет обозначаться через G_0. Специальный остовный подграф графа G_0 есть G_1'; в этом графе строится альтернирующее дерево, как это делалось в алгоритме для ЗНПС. Когда, наконец, образуется и срезается первый цветок, сам граф будет обозначаться через G_1, а его специальный остовный подграф — через G_2'. После срезания f цветков граф будет обозначен символом G_f, а его специальный остовный подграф — G_{f+1}'.

Общий шаг алгоритма таков. Пусть очередной граф (после срезания некоторого числа цветков) будет G_{f-1}, а его специальный остовный подграф — G_f'. В G_f' строится альтернирующее дерево — с помощью алгоритма ЗНПС — до тех пор, пока не произойдет одно из трех:

А) на дереве появится цветок;

Б) дерево станет аугментальным;

В) дерево станет венгерским.

Случай А. В этом случае цветок срезается, получается новый граф G_f и его специальный остовный подграф G_{f+1}'. Как отмечалось выше, срезание цветка приводит к дереву T с корректной структурой альтернирующего дерева, так что T можно сохранить и продолжать процесс роста дерева.

Случай Б. В этом случае, как уже объяснялось для ЗНПС, получается улучшенное паросочетание — с бо́льшим числом ребер.

Дерево T следует отбросить [1]) и, взяв в графе G'_j оставшуюся экспонированную (в G'_f) вершину, приступить к построению нового дерева с корнем в этой вершине. Здесь следует заметить, что как только T отброшено и в G'_j «начинает расти новое дерево», некоторые из псевдовершин в G'_j (образованные после срезания более ранних цветков, давших графы $G_0, G_1, \ldots, G_{f-1}$) могут оказаться помеченными как *внутренние* в новом дереве.

Случай В. В этом случае вектор весов $[\pi_i, \lambda_r]$ для текущего графа G_{f-1} изменяется так, что возникает новый специальный остовный подграф, отличный от G'_f. Изменения в $[\pi_i, \lambda_r]$ делаются так, чтобы

(а) новый граф G'_f продолжал удовлетворять сделанным ранее предположениям, т. е. что λ_r отлично от нуля лишь для множеств вершин, соответствующих псевдовершинам из G_{f-1}, и что равенство выполняется для всех ребер или цветков, которые после срезания дали псевдовершины в G_{f-1};

(б) текущее альтернирующее дерево в старом графе G'_f можно было строить дальше, или на нем возникнет цветок, или оно станет аугментальным — с использованием новых ребер, входящих в новый граф G'_f;

либо некоторая псевдовершина в текущем альтернирующем дереве, помеченная как внутренняя, перестанет быть таковой;

либо будет доказано, что в G_0 нет совершенного паросочетания.

Изменения в векторе весов $[\pi_i, \lambda_r]$ производятся так. Для ребер $a_j = (x_i, x_k)$ из G_{f-1}, не принадлежащих G'_f, одна концевая вершина которых принадлежит текущему альтернирующему дереву T и помечена как внешняя, а другая концевая вершина не лежит в T, находим

$$\Delta_1 = \min_{a_j} [\pi_i + \pi_k - c_j]. \tag{12.19}$$

Для ребер $a_j = (x_i, x_k)$ из G_{f-1}, не принадлежащих G'_f, обе концевые вершины которых лежат в T и обе помечены как внешние, находим

$$\Delta_2 = \frac{1}{2} \min_{a_j} [\pi_i + \pi_k - c_j]. \tag{12.20}$$

Для множеств S_r вершин из G, образующих крайние псевдовершины в T и помеченных как внутренние, находим

$$\Delta_3 = \frac{1}{2} \min_{S_r} [\lambda_r]. \tag{12.21}$$

Берем

$$\Delta = \min [\Delta_1, \Delta_2, \Delta_3]. \tag{12.22}$$

[1]) См. примечание 2 на стр. 382.— *Прим. ред.*

Затем следующим образом изменяем вектор весов $[\pi_i, \lambda_r]$.

Для каждой вершины x_i из G, являющейся внешней вершиной дерева T или содержащейся в псевдовершине из T, помеченной как внешняя, уменьшаем π_i до $\pi_i - \Delta$.

Для каждой вершины x_i из G, являющейся внутренней вершиной дерева T или содержащейся в псевдовершине из T, помеченной как внутренняя, увеличиваем π_i до $\pi_i + \Delta$.

Для каждого множества S_r вершин из G, образующих крайнюю псевдовершину в T, помеченную как внешнюю, увеличиваем λ_r до $\lambda_r + 2\Delta$.

Для каждого множества вершин S_r из G, образующих крайнюю псевдовершину из T, помеченную как внутреннюю, уменьшаем λ_r до $\lambda_r - 2\Delta$.

Безотносительно к значению Δ все ребра из старого графа G'_r, образующие текущее дерево T, остаются в новом графе G'_r, так как веса π_i всех внутренних и внешних вершин дерева T увеличиваются и уменьшаются на одну и ту же величину, а значит равенство в выражении (12.17) по-прежнему сохраняется. Равенство в выражении (12.17) продолжает сохраняться и для всех ребер тех цветков, которые были срезаны, образовав псевдовершины графа G_{f-1}. Таким образом, если псевдовершина из G_{f-1} была помечена как внешняя в альтернирующем дереве T в старом графе G'_f, то для некоторого ребра $a_j = (x_i, x_k)$, входящего в цветок, после срезания которого образовалась эта псевдовершина, веса π_i и π_k уменьшатся на Δ. Но λ_r для множества вершин S_r, соответствующего этой псевдовершине, увеличится на 2Δ, так что равенство в выражении (12.17) по-прежнему сохранится для ребра a_j. Аналогично обстоит дело для ребер тех цветков, которые «порождают» псевдовершины в G_{f-1}, помеченные как внутренние вершины в дереве T в старом графе G'_f.

Предположим теперь, что в (12.22) $\Delta = \Delta_1$ и что ребром, давшим это значение Δ_1 в (12.19), является a_j^*. После Δ — изменения вектора весов $[\pi_i, \lambda_r]$ — ребро a_j^* будет удовлетворять соотношению (12.17), причем имеет место равенство, так что это ребро будет входить в новый граф G'_r. Таким образом, используя a_j^*, можно дерево T (которое, как отмечалось выше, по-прежнему будет альтернирующим в новом графе G'_r) либо расширить, либо сделать аугментальным в зависимости от того, является ли концевая вершина ребра a_j^*, не принадлежащая дереву T, экспонированной или нет.

Если в (12.22) $\Delta = \Delta_2$ и a_j^* — ребро, дающее это значение в (12.20), то добавление a_j^* к дереву T приведет к образованию цветка.

Наконец, если в (12.22) $\Delta = \Delta_3$, то вычитание величины 2Δ из λ_r в процессе изменения вектора весов $[\pi_i, \lambda_r]$ сделает в соответствии с (12.21) некоторое λ_r (скажем, λ^*) равным нулю. Пусть

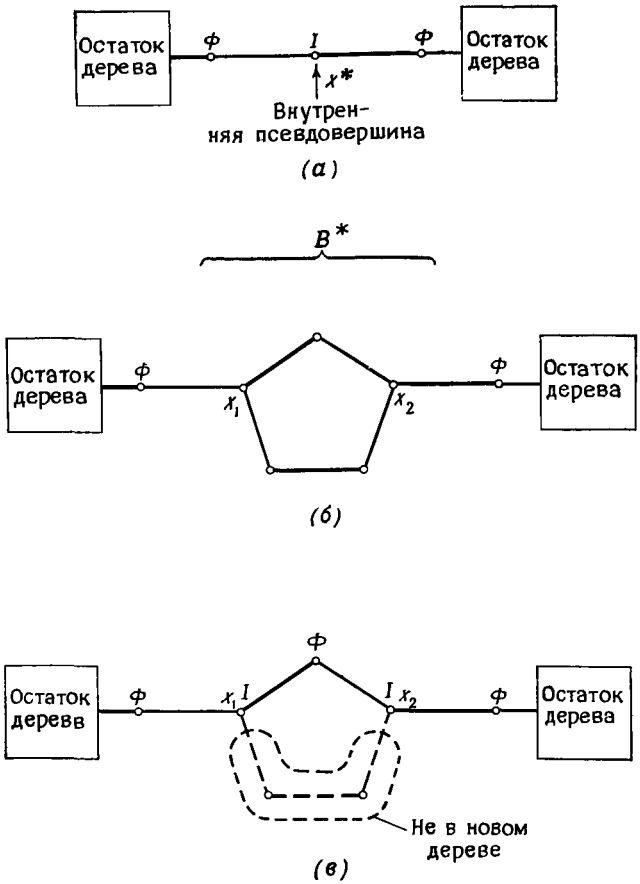


Рис. 12.15. (б) Внутренняя псевдовершина распускается и становится крайним цветком B^*.
(в) Образование нового дерева.

x^* — внутренняя псевдовершина текущего дерева T, соответствующая λ_r^*, и B^* — крайний цветок, давший после срезания x^*. Псевдовершина x^* графа G_{f-1} может теперь расцвести, образовав цветок B^*, и это даст другой граф G_{f-2}. (Заметим, однако, что этот граф G_{f-2} не будет тем же самым графом, из которого был получен граф G_{f-1} после срезания цветка, так как распустившийся цветок, превративший G_{f-1} в G_{f-2}, не будет тем же самым крайним цветком, который после срезания дал G_{f-1}.) Пусть G'_{f-1} — специальный остовный подграф нового графа G_{f-2}. Так как для всех ребер из B^* в (12.17) имеет место равенство, то очевидно,

что все эти ребра входят и в G'_{f-1}. Если B^* состоит, скажем, из $2q+1$ ребер, то в B^* легко получить паросочетание из q ребер, как это показано на рис. 12.15. После того как цветок B^* распустился, он разбивает текущее альтернирующее дерево на две компоненты, одна из которых «касается» цветка в вершине x_1, а другая — в вершине x_2. Если использовать только ребра из B^*, то можно указать две цепи (одну с четным и другую с нечетным числом ребер) между вершинами x_1 и x_2. Добавляя ту цепь, у которой четное число ребер, к текущему дереву T, связываем T снова в одно дерево (см. рис. 12.15 (в)). Новое альтернирующее дерево можно затем расширять до тех пор, пока опять не возникнут приведенные выше случаи А, Б или В.

Алгоритм завершает работу, когда на некотором этапе f в графе G'_f будет получено совершенное паросочетание. Соответствующий граф G_{f-1} расширяется до начального графа G_0 путем распускания цветков в порядке, обратном порядку их срезания, и построения паросочетаний в каждом распустившемся цветке — так же, как на шаге 8 алгоритма ЗНПС из разд. 2.4.1.

Когда получено совершенное паросочетание, оно будет максимальным совершенным паросочетанием, так как тогда мы имеем решение (т. е. вектор $[\xi_j]$) и двойственный вектор весов $[\pi_i, \lambda_r]$, удовлетворяющий ограничениям (12.17) и (12.18), а также сформулированным выше условиям (i) и (ii).

Альтернативным случаем остановки является случай В, когда не существует никаких ограничений (12.19), (12.20) и (12.21). В этом случае Δ может быть взято как угодно большим, и совершенно очевидно, что в G не существует никакого совершенного паросочетания.

Приведенный выше метод решения ЗМП был предложен Эдмондсом [11]. Он является эффективным алгоритмом (см. разд. 3.3) и служит также конструктивным доказательством теоремы 5.

3.1.1. Описание алгоритма

Присвоение начальных значений

Шаг 1. Присвоить вершинам графа G_0 веса π_i, так чтобы для всех ребер из G_0 было выполнено соотношение (12.17), когда все $\lambda_r = 0$. Взять $G = G_0$.

Формирование графа G'

Шаг 2. Образовать специальный остовный подграф G' текущего графа G.

Построение дерева

Шаг 3. Если в G' не существует никакого альтернирующего дерева T, то строить новое дерево с корнем в некоторой экспониро-

ванной вершине из G'. Если экспонированных вершин нет, перейти к шагу 8. Если в G' уже существует альтернирующее дерево, то продолжать его построение. Если на T обнаружится цветок, то перейти к шагу 4. Если дерево T станет аугментальным, то перейти к шагу 5.

Если дерево T станет венгерским, то перейти к шагу 6.

Цветки на дереве

Шаг 4. Срезать цветки и образовать псевдовершину. Обозначим получившийся граф через G, его специальный остовный подграф через G', а оставшееся дерево через T. Перейти к шагу 3.

Аугментальное дерево

Шаг 5. Улучшить текущее паросочетание, «поменяв» вдоль аугментальной цепи ребра, принадлежащие паросочетанию и ему не принадлежащие. «Отбросить» дерево T и перейти к шагу 3.

Венгерское дерево

Шаг 6. Вычислить Δ_1, Δ_2, Δ_3 и Δ по формулам (12.19) — (12.22). Если нет никаких ограничений (12.19) — (12.21), то остановиться; в графе нет совершенного паросочетания. В противном случае изменить вектор весов $[\pi_i, \lambda_r]$, как описано выше. Если $\Delta = \Delta_1$ или $\Delta = \Delta_2$, то сохранить текущее дерево T и перейти к шагу 2. Если $\Delta = \Delta_3$, то перейти к шагу 7.

Шаг 7. «Распустить» псевдовершину, давшую Δ_3 в соотношении (12.21), в цветок B. Обозначим получившийся граф через G, а его специальный остовный подграф через G'. Построить в B совершенное паросочетание. Перестроить альтернирующее дерево, добавляя к ребрам из T необходимую цепь из B, и обозначить это дерево символом T. Перейти к шагу 3.

Конец

Шаг 8. Распустить все цветки в обратном порядке (к первоначальному порядку срезания цветков), находя совершенное паросочетание после каждого распускания.

3 2. Пример

Мы хотим найти совершенное паросочетание для графа G_0, изображенного на рис. 12.16. Начнем с приписывания вершинам x_i, $i = 1, \ldots, 12$, весов π_i, задаваемых вектором $[\pi_i]^t = (9, 2, 11, 7, 3, 4, 8, 6, 6, 5, 3, 4)$, и полагая все $\lambda_r = 0$. Веса π_i выбраны нами довольно произвольно.

Специальный остовный подграф G_1' графа G_0 изображен на рис. 12.17. Пусть в качестве корня выбрана вершина x_1, и в G_1'

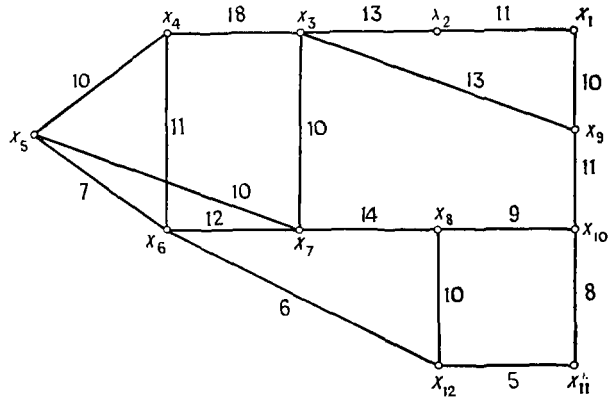


Рис. 12.16. Граф G_0 из примера 3.2.

на шаге 3 алгоритма строится альтернирующее дерево. На шаге 5 сразу же в паросочетание отбирается ребро (x_1, x_2). Выбирая вершины x_3, x_5 и x_7 (в указанном порядке) в качестве корней новых деревьев, немедленно получим аугментальные деревья и приходим к ситуации, в которой ребра в паросочетании показаны жирными линиями на рис. 12.17. Выбрав вершину x_{12} в качестве корня нового дерева, строим это дерево на шаге 3 алгоритма до тех пор, пока не будет достигнуто состояние, изображенное на рис. 12.18, в котором образуется цветок. На шаге 4 этот цветок срезается и образуется псевдовершина x_{b_1}, а рост дерева продолжается на шаге 3, пока не достигается состояние, представленное на рис. 12.19, в котором дерево становится венгерским. Сам граф (после срезания) будет G_1, а его специальный остовный подграф — G_2', как показано соответственно на рис. 12.20 (а) и рис. 12.20 (б).

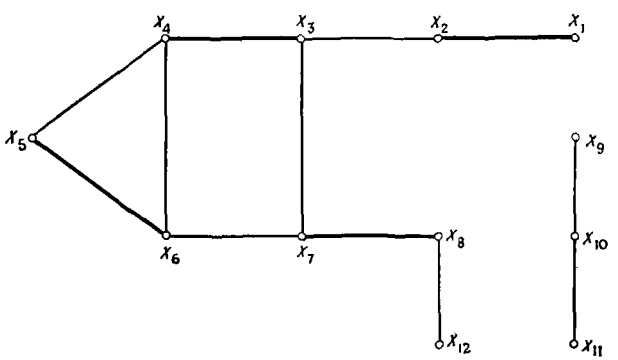


Рис. 12.17. Специальный остовный подграф G_1'.

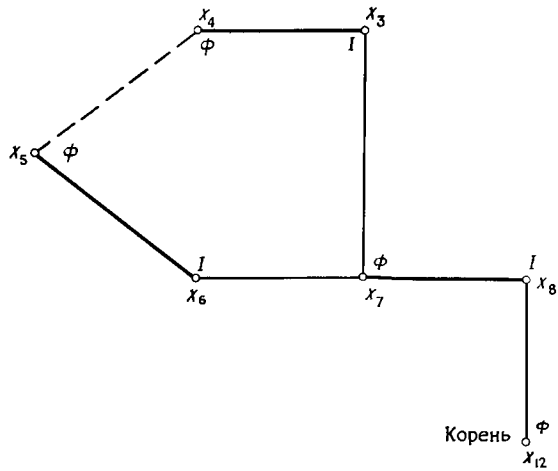


Рис. 12.18. Альтернирующее дерево в G_1'. Корень.

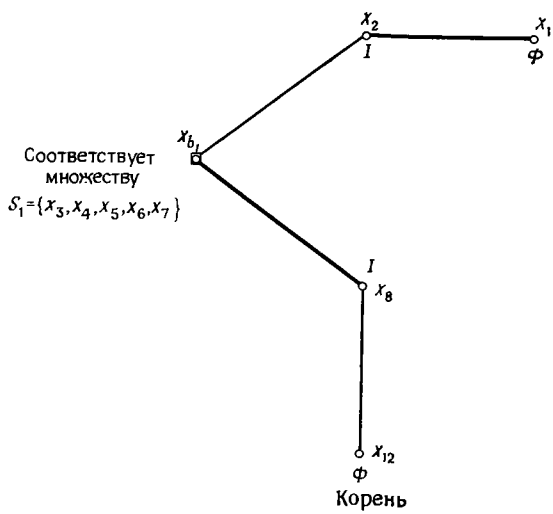


Рис. 12.19. Венгерское дерево в G_1'.

Шаг 6 алгоритма дает

$$\Delta_1 = \min [\underbrace{9+6-10}_{\text{ребро } (x_1, x_9)},\ \underbrace{11+6-13}_{\text{ребро } (x_3, x_8)},\ \underbrace{4+3-5}_{\text{ребро } (x_{12}, x_{11})}] =$$

$= 2$ соответствует ребру (x_{12}, x_{11}),

$\Delta_2 = \frac{1}{2}[4+4-6] = 1$ соответствует ребру (x_6, x_{12}),

$\Delta_3 = \infty$ ограничение отсутствует.

Таким образом, $\Delta = \min [2, 1, \infty] = 1$.

После изменения весов новый вектор π принимает вид $[\pi_i]^2 = (8, 3, 10, 6, 2, 3, 7, 7, 6, 5, 3, 3)$, а вес λ_1, соответствующий нечетному множеству вершин $S_1 = \{x_3, x_4, x_5, x_6, x_7\}$, равен 2.

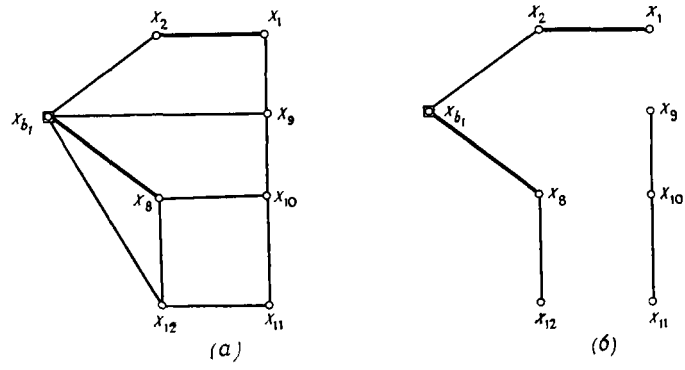


Рис. 12.20. (а) Граф G_1. (б) Граф G_2'.

Для нового вектора весов $[\pi_i, \lambda_1]$ новый специальный остовный подграф G_2' графа G_1 изображен на рис. 12.21; ребро (x_{12}, x_{b_1}) входит в старый подграф G_2'.

Так как дерево на рис. 12.19 сохраняется, то новое вошедшее ребро приводит к образованию цветка, состоящего из нечетного цикла (x_{12}, x_8, x_{b_1}). Это сразу же обнаруживается на шаге 3, а на шаге 4 цветок срезается; возникает псевдовершина x_{b_2} и получается новое дерево, показанное на рис. 12.22. Сам граф теперь будет G_2, а его специальный остовный подграф — G_3'; эти графы представлены соответственно на рис. 12.23а и рис. 12.23б.

Дерево, изображенное на рис. 12.22, является венгерским деревом в G_3', и на шаге 6 вычисляется новое значение Δ для изменения

вектора весов:

$$\Delta_1 = \min [\underbrace{8+6-10}_{\text{ребро } (x_1, x_9)},\ \underbrace{10+6-13}_{\text{ребро } (x_3, x_9)},\ \underbrace{7+5-9}_{\text{ребро } (x_8, x_{10})},\ \underbrace{3+3-5}_{\text{ребро } (x_{12}, x_{11})}] =$$

$= 1$ соответствует ребру (x_{12}, x_{11}),

$\Delta_2 = \infty$ и $\Delta_3 = \infty$.

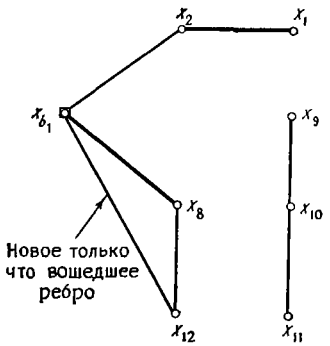


Рис. 12.21. Новый специальный остовный подграф G_2' для графа G_1 из рис. 12.20(а).

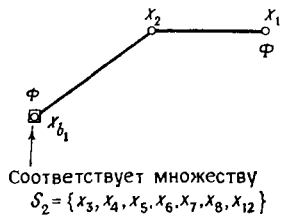


Рис. 12.22. Альтернирующее дерево в G_2' из рис. 12.21.

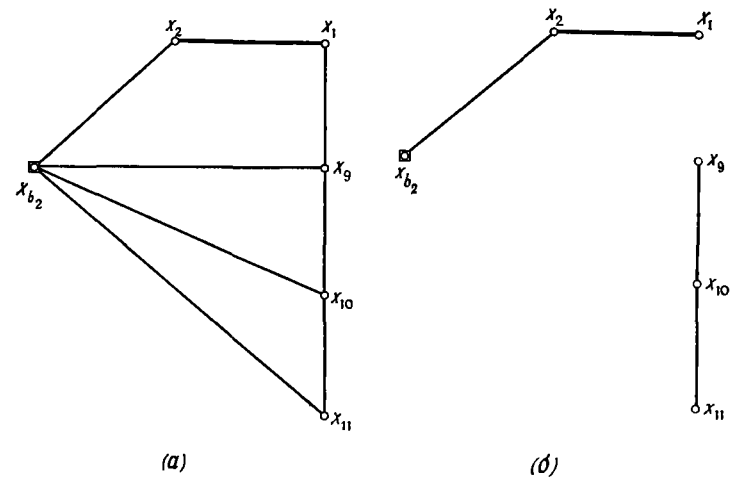


Рис. 12.23. (а) Граф G_2. (б) Граф G_3'.

Следовательно, $\Delta = 1$. Используя это значение Δ, получаем $[\pi_i]^3 = (7, 4, 9, 5, 1, 2, 6, 6, 6, 5, 3, 2)$, а вес λ_2, соответствующий нечетному множеству вершин $S_0 = \{x_3, x_4, x_5, x_6, x_7, x_8, x_{12}\}$, равен $\lambda_2 = 2$.

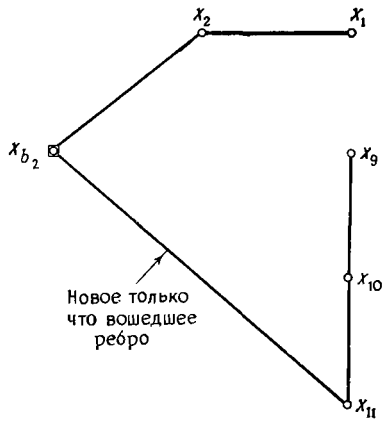


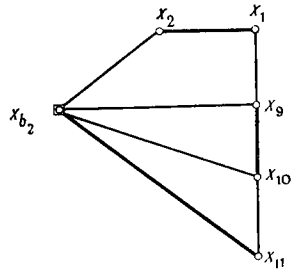
Рис. 12.24. Новый специальный остовный подграф G_3' для графа G_2 из рис. 12.23(а).

Для этих новых весов $[\pi_i, \lambda_r]$ новый специальный остовный подграф графа G_2 изображен на рис. 12.24. После введения нового ребра (x_{12}, x_{11}) альтернирующее дерево на рис.12.22 становится аугментальным, так что новое ребро входит в паросочетание. Это дерево «отбрасывается», и строится новое дерево с корнем в одной из оставшихся экспонированных вершин графа G_3', т. е. в вершине x_9 или x_{10}. В любом случае ребро (x_9, x_{10}) сразу же попадает в паросочетание и получается совершенное паросочетание. Это совершенное паросочетание графа G_2 показано на рис. 12.25 (а). Чтобы найти соответствующее паросочетание в G_0, мы сначала распустим цветок b_2 и построим в нем совершенное паросочетание, как показано на рис. 12.25 (б), а затем распустим цветок b_1 и построим в нем совершенное паросочетание (см. рис. 12.25 (в)). На этом рисунке жирными линиями изображено максимальное совершенное паросочетание графа G_0; вес этого паросочетания равен 66. Так как в соответствии с теорией линейного программирования максимум величины z из (12.11) равен минимуму величины u из (12.16), то можно сделать проверку. Поскольку $|S_1| = 2q_1 + 1 = 5$ и $|S_2| = 2q_2 + 1 = 7$, то $q_1 = 2$ и $q_2 = 3$, так что

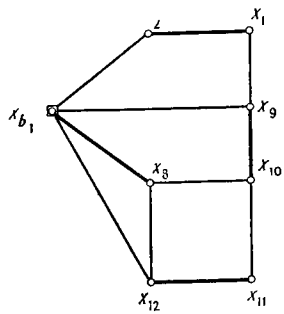
$$u = (7+4+9+5+1+2+6+6+6+5+3+2)+ \\ + (2\times 2+3\times 2) = 66$$

3.3. Некоторые вычислительные результаты

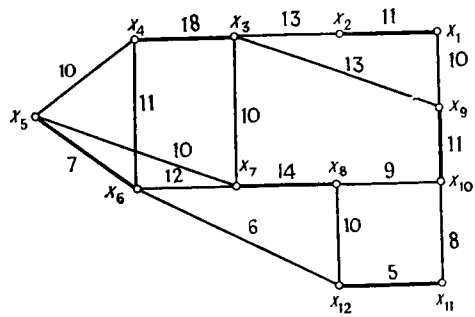
Алгоритм для ЗНП, а следовательно, и для ЗМП, является хорошим алгоритмом в том смысле, что необходимое для него число операций является полиномиальной функцией от числа n вершин графа. Для ЗНП требуемое для вычислений время растет как $O[n^4]$, хотя на практике этот рост значительно медлен-



(а)



(б)



(в)

Рис. 12.25. (а) Совершенное паросочетание в G_2. (б) Паросочетание с распустившейся вершиной b_2. (в) Максимальное паросочетание.

нее [30]. Программа для вычислительной машины, приведенная в [30], дает зависимость вида $O[n^{2,7}]$, а другая программа [29] — зависимость вида $O[n^{2,3}]$. Эдмондс и Джонсон [29] привели типичный результат для задачи о паросочетании (на самом деле — для общей задачи, обсуждавшейся в разд. 5, с $\delta_i = 1$ или 2 и $1 \leqslant c_j \leqslant 10$) с 300 вершинами и 1500 ребрами, потребовавшей для решения около 30 секунд на ИБМ 360/91.

4. Задача о назначениях

В этом разделе мы рассмотрим частный случай паросочетаний — в двудольных графах [9, 18, 25]. Задан двудольный граф $G = (X^a \cup X^b, A)$, где X^a и X^b — независимые множества вершин и каждое ребро $a_j = (x_i, x_k) \in A$ имеет $x_i \in X^a$, $x_k \in X^b$ и вес c_j. Мы хотим найти совершенное паросочетание графа G с максимальным (или минимальным) весом. Минимизационный вариант этой задачи хорошо известен в литературе как *задача о назначениях* (ЗН), и он без потери общности часто обсуждается в связи с полными двудольными графами.

Пусть $|X^a| = |X^b| = n$. Рассматривая полный двудольный граф $K_{n,n}$, легко видеть, что задача о паросочетании для этого специального случая может быть решена с помощью методов, предназначенных для исследования потоков в сетях (из гл. 11), и поэтому нет необходимости в применении общего алгоритма паросочетания, изложенного в разд. 3 настоящей главы. Пусть добавлен искусственный источник s с дугами (s, x_i), $\forall x_i \in X^a$, имеющими единичную пропускную способность и нулевую стоимость (вес), и искусственный сток t с дугами (x_k, t), $\forall\, x_k \in X^b$, также имеющими единичную пропускную способность и нулевую стоимость (вес). Если рассматривать ребра из A как ориентированные, т. е. как дуги (x_i, x_k) с единичными пропускными способностями, то максимальный поток с минимальной стоимостью (весом) от s к t будет иметь значение n, а по этим дугам будет «проходить» ненулевой поток (со значением 1) [1]) с минимальной суммарной стоимостью (весом). Эти дуги из A будут тогда образовывать в G совершенное паросочетание с минимальной стоимостью (весом). Очевидно, что в случае такой специальной структуры графа многие шаги общего алгоритма нахождения потока минимальной стоимости (веса) могут быть упрощены с целью получения более эффективной процедуры для ЗН.

---

[1]) Отметим, что в соответствии с гл. 11, если все пропускные способности — целые, то потоки по дугам тоже будут целые, т. е. в нашем случае 0 или 1.

С другой стороны, ЗН можно рассматривать как частный случай ЗМП и решать ее с помощью специализированной версии алгоритма из разд. 3.1.1. Главная трудность, связанная с этим общим алгоритмом — образование и срезание цветков,— в случае ЗН отсутствует, так как в двудольном графе не может существовать циклов с нечетным числом ребер. Так как обычно принято рассматривать ЗН как задачу минимизации, то мы последуем этому обычаю и опишем минимизационный вариант алгоритма из разд. 3.1.1, применимый только к двудольным графам. Изменения, которые нужно сделать в случае задачи максимизации, тривиальны.

4.1. Алгоритм для ЗН (случай минимизации)

Описанный здесь алгоритм был предложен Кёнигом [22], Эгервари [15] и Куном [23, 24] и известен как *венгерский алгоритм*.

4.1.1. Описание алгоритма

Присвоение начальных значений

Шаг 1. Вершинам из множеств X^a и X^b графа G присвоить веса π_i^a и π_k^b соответственно, причем так, чтобы для любого ребра $a_j = (x_i, x_k)$ выполнялось неравенство $\pi_i^a + \pi_k^b \leqslant c_j$.

Формирование графа G'

Шаг 2. Построить специальный остовный подграф G' графа G с данными текущими весами $[\pi]$.

Построение дерева

Шаг 3. Если в G' нет никакого альтернирующего дерева T, то «выращивать» его, взяв в качестве корня некоторую экспонированную вершину x_i графа G', принадлежащую множеству X^a; если экспонированных вершин нет, то перейти к шагу 7. Если альтернирующее дерево уже существует, то продолжать «выращивать» его. Если дерево T окажется аугментальным, то перейти к шагу 4, а если венгерским — к шагу 5.

Аугментальное дерево

Шаг 4. Улучшить текущее паросочетание, «взаимно поменяв» (вдоль аугментальной цепи) ребра, принадлежащие ему и ему не принадлежащие.
«Отбросить» дерево T и перейти к шагу 3.

Венгерское дерево

Шаг 5. Для ребер $a_j = (x_i, x_k)$, не лежащих в G', одна концевая вершина которых принадлежит текущему дереву T и помечена

как внешняя, а другая концевая вершина не принадлежит T, найти

$$\Delta = \min_{a_j} [c_j - \pi_i^a - \pi_k^b].$$

Шаг 6. Для каждой вершины x_i графа G, принадлежащей X^a и являющейся внешней вершиной дерева T, поменять π_i^a на $\pi_i^a + \Delta$, а для каждой вершины x_k графа G, принадлежащей X^b и являющейся внутренней вершиной дерева T, заменить π_k^b на $\pi_k^b - \Delta$.
Сохранить дерево T и перейти к шагу 2.

Конец

Шаг 7. Текущее совершенное паросочетание является минимальным.

4.1.2. Матричная форма алгоритма

Часто, когда граф является полным, операции вышеприведенного алгоритма реализуются на $(n \times n)$-матрице $\mathbf{C}$, строки которой соответствуют вершинам из X^a, а столбцы — вершинам из X^b. Начальные элементы c_{ik} являются стоимостями (весами) ребер (x_i, x_k).

На шаге 1 алгоритма начальный вес π_i^a вершины $x_i \in X^a$ берется равным наименьшему элементу строки i. Вес π_i^a вычитается из всех элементов строки i, и так делается для всех строк. Вес π_k^b вершины $x_k \in X^b$ берется равным наименьшему элементу в столбце k получившейся матрицы. Затем π_k^b вычитается из всех элементов столбца k, и это делается для всех столбцов, в результате чего получается новая матрица $\mathbf{C}'$ с $c'_{ik} = c_{ik} - \pi_i^a - \pi_k^b \geqslant \geqslant 0$. Матрица $\mathbf{C}'$ называется *редуцированной матрицей*. Тогда остовным подграфом G' будет $(n \times n)$-двудольный граф с дугами (x_i, x_k), соединяющими только те вершины x_i и x_k, для которых $c'_{ik} = 0$. Совершенное паросочетание графа G' будет соответствовать такому выбору нулевых элементов в матрице $\mathbf{C}'$, когда в каждой строке и каждом столбце выбрано ровно по одному нулевому элементу $\mathbf{C}'$.

Построение альтернирующего дерева в этом получившемся остовном подграфе G' происходит с помощью следующей процедуры расстановки пометок в строках и столбцах матрицы $\mathbf{C}'$. Начнем построение произвольного паросочетания в G', отмечая нулевые элементы в $\mathbf{C}'$ так, чтобы никакие два отмеченные нуля не находились в одном и том же столбце или строке. (Если мы отметили нуль в позиции (i, k), то это означает, что ребро (x_i, x_k) входит в паросочетание, и наоборот.)

Найдем строку s без отмеченных элементов и сделаем пометку $p_s = 0$. Если такой строки нет, то текущее паросочетание будет минимальным совершенным паросочетанием. (Вершина x_s — если такая существует — соответствует теперь экспонированной вершине, выбранной в качестве корня альтернирующего дерева, а p_i являются эквивалентами пометок, используемых для «хранения» этого дерева [1]).) Это надо понимать так: дерево рассматривается как некоторая древовидность с корнем, причем если дуга (x_α, x_β) входит в эту древовидность, то «хранится равенство» $p_\beta = x_\alpha$. Для корня дерева мы берем $p_s = 0$. (См. разд. 2.2.1, гл. 7.) Начинаем с такой ситуации, когда есть пометка $p_s = 0$ и когда все другие строки и столбцы не помечены.

(i) Пометить каждый непомеченный столбец k, содержащий неотмеченный 0 в помеченной строке i, меткой $p_k = i$. Если существует несколько возможностей, взять любую.

(ii) Пометить каждую непомеченную строку i, содержащую отмеченный 0 в помеченном столбце k, меткой $p_i = k$.

Повторять процесс расстановки пометок (в соответствии с правилами (i) и (ii)). Эти две операции неявно строят в G' дерево, причем это дерево дается эквивалентами пометок. Стоит заметить, что все «внешние» вершины этого дерева соответствуют строкам из $\mathbf{C}'$, а все «внутренние»— столбцам. Применение операций (i) и (ii) продолжается до тех пор, пока не будет помечен столбец t без отмеченных элементов (скажем, меткой p_t) или процесс расстановки пометок нельзя будет продолжать. В первом случае будет найдена аугментальная цепь t, p_t, $p_{t'}$, $p_{t''}$, . . ., s, где $t' = p_t$, $t'' = p_{t'}$ и т. д. Элементы 0 (нули) в позициях (t, p_t), $(p_t, p_{t'})$, $(p_{t'}, p_{t''})$, . . ., $(\cdot, s)$ матрицы $\mathbf{C}'$ попеременно отмечаются или нет для образования лучшего паросочетания. Пометки p_i стираются, и процесс продолжается. Во втором случае обнаруживается венгерское дерево. Если I^+ и K^+ — множества всех помеченных, а I^- и K^- — всех непомеченных строк и столбцов соответственно, то находим

$$\Delta = \min_{\substack{i \in I^+ \\ k \in K^-}} [c'_{ik}].$$

(Заметим, что $c'_{ik} = c_{ik} - \pi_i^a - \pi_k^b$.)

Обновление. $\pi_i = \pi_i + \Delta$ для $i \in I^+$, $\pi_k = \pi_k - \Delta$ для $k \in K^+$, $c'_{ik} = c'_{ik} - \Delta$ для $i \in I^+$ и $k \in K^-$, $c'_{ik} = c'_{ik} + \Delta$ для $i \in I^-$ и $k \in K^+$, в остальных случаях c'_{ik} не изменяются. Операции (i) и (ii) теперь снова применяются к новой матрице $\mathbf{C}'$ и т. д.

---

[1]) См. примечание 2 на стр. 382.— *Прим. ред.*

4.2. Пример

Решим ЗН для полного двудольного графа $G = K_{8,8}$, где матрица C стоимостей (весов) такая:

$\mathbf{C} =$

	1	2	3	4	5	6	7	8
1	13	21	20	12	8	26	22	11
2	12	36	25	41	40	11	4	8
3	35	32	13	36	26	21	13	37
4	34	54	7	8	12	22	11	40
5	21	6	45	18	24	34	12	48
6	42	19	39	15	14	16	28	46
7	16	34	38	3	34	40	22	24
8	26	20	5	17	45	31	37	43

Редуцированной матрицей C′ будет (8×8)-матрица, приведенная ниже, где векторы слева от матрицы и вверху — весовые векторы $[\pi_i]$ и $[\pi_k]$ соответственно.

$\pi_i \downarrow$ \ $\pi_k \rightarrow$		5	0	−5	−4	0	2	−1	3	
		1	2	3	4	5	6	7	8	
8	1	0*	13	17	8	0	16	15	0	
5	2	2	31	25	40	35	4	0	0*	
14	3	16	18	4	26	12	5	0*	20	7
12	4	17	42	[0*]	0	[0]	8	0	25	3
6	5	10	0*	44	16	18	26	7	39	
14	6	23	5	30	5	[0*]	[0]	15	29	5
7	7	4	27	36	0*	27	31	16	14	4
10	8	11	10	[0]	11	35	19	28	30	0
				8	4	4	6	4		← p_k

↑ p_i

Начнем со случайно выбранного паросочетания, т. е. случайно выберем некоторое множество нулей, никакие два элемента в котором не лежат в одной строке или столбце. Выбранные нули отмечены звездочкой. В строке 4 нет отмеченных элементов, и она получает пометку $p_{\text{стр }4} = 0$. Следующие пометки располагаются в таком порядке $p_{\text{столб }3} = 4$, $p_{\text{стр }3} = 3$, $p_{\text{столб }7} = 3$, $p_{\text{стр }2} = 7$. Окончательные векторы пометок $[p_i]$ и $[p_k]$ показаны справа от матрицы $\mathbf{C}'$ и под ней.

Дальше помечать нечего. Теперь мы имеем $I^+ = \{2, 3, 4\}$, $K^+ = \{3, 7\}$, $I^- = \{1, 5, 6, 7, 8\}$ и $K^- = \{1, 2, 4, 5, 6, 8\}$. После вычисления Δ, которое равно 1, и после изменения весов $[\pi_i]$ и $[\pi_k]$ получаем новую матрицу $\mathbf{C}' = [c_{ik} - \pi_i - \pi_k]$, изображенную ниже:

$\pi_i \downarrow$		$\pi_k \rightarrow$ 5	0	−1	0	0	2	−1	3	
		1	2	3	4	5	6	7	8	
8	1	0*	13	13	4	0	16	15	0	
5	2	2	31	21	36	35	4	0	0*	
14	3	16	18	0	22	12	5	0*	20	
8	4	21	46	0*	0	4	12	4	29	3
6	5	10	0*	40	12	18	26	7	39	
14	6	23	5	26	1	0*	0	15	29	
3	7	8	31	36	0*	31	35	20	18	4
6	8	15	14	0	11	39	23	32	34	0
										$\uparrow p_i$
				8	4					$\leftarrow p_k$

Расстановка пометок продолжается, так что $p_{\text{столб }4} = 4$ и $p_{\text{столб }8} = 2$. Так как столбец 8 не имеет отмеченных нулей, то найдена аугментальная цепь, показанная стрелками в вышеприведенной матрице. «Меняя» отмеченные и неотмеченные нули вдоль этой цепи, получаем новое, улучшенное, паросочетание.

При второй итерации пометки стираются и в качестве строки без отмеченных элементов выбирается строка 8, которая отмечается как $p_{\text{стр }8} = 0$. Больше пометок расставить нельзя, Δ оказы-

вается равным 1, $\pi_{\text{стр }8}$ заменяется на 6 и новая матрица C′ дана ниже. (Заметим, что при второй итерации венгерское дерево является на самом деле единственной изолированной — в терминах текущего графа G' — вершиной, соответствующей строке 8.) Затем процесс расстановки пометок продолжается и получаются векторы $[p_i]$ и $[p_k]$, приведенные ниже.

$\pi_k \rightarrow$	5	0	−1	0	0	2	−1	3

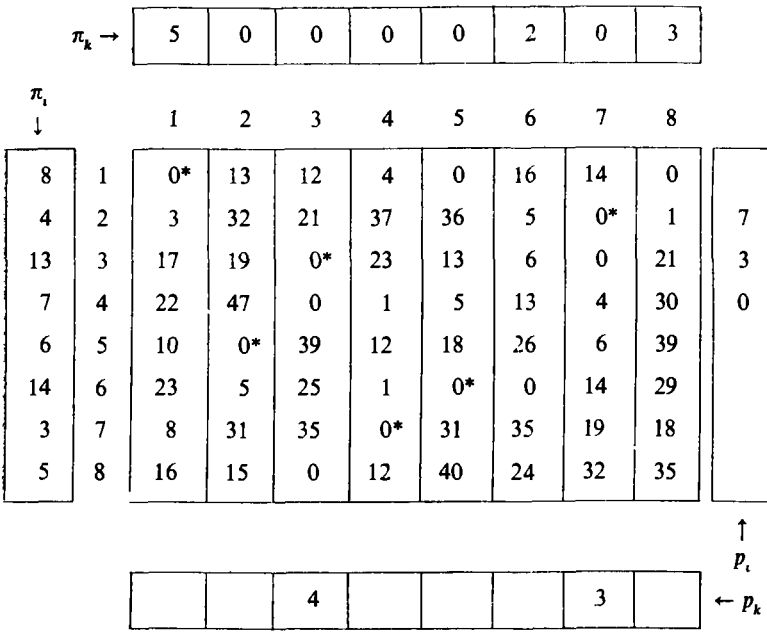
$\pi_i \downarrow$		1	2	3	4	5	6	7	8	
8	1	0*	13	13	4	0	16	15	0	
5	2	2	31	21	36	35	4	[0*]	[0]	7
14	3	16	18	[0*]	22	12	5	[0]	20	3
8	4	21	46	[0]	0	4	12	4	29	0
6	5	10	0*	40	12	18	26	7	39	
14	6	23	5	26	1	0*	0	15	29	
3	7	8	31	36	0*	31	35	20	18	
5	8	16	15	1	12	40	24	33	35	

$\uparrow p_i$

		4	4			3	2	$\leftarrow p_k$

Опять помечать больше нечего, Δ оказывается равным 4, веса $[\pi_i]$ и $[\pi_k]$ изменяются и ниже дается новая матрица C′ (см. стр. 411).

Продолжается процесс расстановки пометок, и эти пометки (с учетом порядка их расстановки) таковы: $p_{\text{столб }5} = p_{\text{столб }7} = 4$, $p_{\text{стр }6} = 5$, $p_{\text{стр }3} = 7$, $p_{\text{столб }6} = 6$. Так как столбец 6 не имеет отмеченных элементов и он помечен, то найдена аугментальная цепь, показанная стрелками в матрице C′. «Изменяя» вдоль этой цепи отмеченные и не отмеченные нули, получаем совершенное паросочетание, являющееся поэтому паросочетанием с минимальной стоимостью (весом). Ребра этого паросочетания таковы: (1, 1), (2, 8), (3, 7), (4, 5), (5, 2), (6, 6), (7, 4) и (8, 3), а полная стоимость (вес) равна 76 (для проверки заметим, что $\sum_i \pi_i + \sum_k \pi_k = 76$).



5. Общая задача построения остовного подграфа с предписанными степенями

Для заданного графа $G = (X, A)$, дугам которого приписаны тоимости, *общая задача* была сформулирована во введении как адача нахождения остовного подграфа G_p^*, по отношению к кото-ому степени вершин x_i равны заданным целым числам δ_i и сум-а стоимостей ребер в G_p^* максимальна (или минимальна). Во ввеении отмечалось, что ЗМП является частным случаем этой *общей ɜдачи*. Теперь мы покажем, что общая задача сама может быть ешена как ЗМП с использованием алгоритма из разд. 3.1.1.

Исходя из графа G, построим граф $\hat{G}$. Если требуется, чтобы $_i$ было степенью вершины x_i в графе G, то в $\hat{G}$ будут сопоставлены ершине x_i δ_i вершин $x_i^1, x_i^2, \ldots, x_i^{\delta_i}$. Каждому ребру $a_j = (x_i, _k)$ графа G будут соответствовать в $\hat{G}$ две дополнительные вершины $_j$ и s_j, ребра (x_i^α, r_j), $\alpha = 1, \ldots, \delta_i$, все со стоимостями $\frac{1}{2} c_{ik}$, ебра (s_j, x_k^β), $\beta = 1, \ldots, \delta_k$, со стоимостями $\frac{1}{2} c_{ik}$ и ребро $_j, s_j)$ со стоимостью 0.

Теперь совершенно очевидно, что общая задача для графа G соответствует задаче о назначениях для графа $\hat{G}$. Если ребро (r_j, s_j) лежит в паросочетании $\hat{G}$, то это означает, что ребро (x_i, x_k) не принадлежит графу G_p^*. Если два ребра (x_i^α, r_j) и (s_j, x_k^β) лежат в паросочетании графа $\hat{G}$, то это означает, что ребро (x_i, x_k) принадлежит G_p^*. Существует взаимно однозначное соответствие между паросочетаниями в $\hat{G}$ и остовными подграфами с допустимыми степенями в графе G. Но так как число вершин в $\hat{G}$ намного больше n, вышеприведенное преобразование графа G в $\hat{G}$ с вычислительной точки зрения неэффективно. На самом деле не обязательно строить явно граф $\hat{G}$ и, возможно [13], так модифицировать алгоритм из разд. 3.1.1, что он будет оперировать непосредственно с первоначальным графом G, но все будет относиться к графу $\hat{G}$.

5.1. Транспортная задача

Общая задача для двудольного графа $G = (X^a \cup X^b, A)$ широко известна как транспортная задача (ТЗ) [8, 16, 18, 25, 4, 21] и является обобщением ЗН, обсуждавшейся в разд. 4. Транспортная задача получила свое название из-за следующей ее интерпретации. Имеется N источников (пунктов снабжения) и M стоков (пунктов потребления). Производительность i-го источника снабжения равна α_i, потребление k-го стока равно β_k, причем

$$\sum_{i=1}^{N} \alpha_i = \sum_{k=1}^{M} \beta_k.$$

Требуется найти такую схему транспортировки груза от источников к стокам, которая имеет минимальную стоимость, если стоимость транспортировки единицы груза от i-го источника к k-му стоку равна c_{ik}.

Если α_i и β_k — целые числа, то ТЗ можно также рассматривать как *общую задачу* о *паросочетании* в двудольном графе и сначала построить граф $\hat{G}$, как было объяснено ранее, а затем решить для этого графа ЗН. При альтернативном подходе ТЗ можно рассматривать как задачу о потоке в сети, точно так же, как это делалось для ЗН в разд. 4.

Ввиду практической важности ТЗ мы дадим матричную форму венгерского алгоритма для этой задачи. Этот алгоритм очень похож на алгоритм из разд. 4.1.2 для ЗН и приводится здесь без дальнейших пояснений.

5.1.1. Венгерский алгоритм для ТЗ

Присвоение начальных значений

Шаг 1. Начиная с матрицы стоимостей C, построить редуцированную матрицу C′.

Нахождение аугментальной цепи

Шаг 2. Пометить строку s, для которой $\alpha_s \neq 0$, пометкой $'_s = 0$. Если таких строк нет, перейти к шагу 9.

Шаг 3. Пометить каждый непомеченный столбец k, содержащий 0 в помеченной строке i (т. е. имеющей $c'_{ik} = 0$), пометкой $_k = i$. Если существует несколько возможностей, выбрать любую.

Шаг 4. Пометить каждую непомеченную строку i, содержащую отмеченный элемент в помеченном столбце k, пометкой $_i = k$.

Шаг 5. Повторять шаги 3 и 4 до тех пор, пока либо не будет отмечен столбец t с $\beta_t \neq 0$, в случае чего перейти к шагу 6, либо дальнейшая расстановка пометок невозможна, в случае чего перейти к шагу 8.

Аугментация

Шаг 6. Найти аугментальную цепь P между s и t и взять в качестве ε минимальное из следующих чисел:

$$\min [\alpha_s, \beta_t] \quad \text{и} \quad \min_{\substack{(i,k) \text{ в } P \\ \text{и отмечен}}} [c'_{ik}].$$

Шаг 7. Попеременно прибавлять $+\varepsilon$ и $-\varepsilon$ (именно в таком порядке) ко всем элементам (i, k) в альтернирующей цепи P. Отметить те элементы, к которым ε прибавлялось, и удалить пометки у всех элементов, значение которых после вычитания стало равным нулю.

Уменьшить α_s и β_t на ε. Удалить все пометки и вернуться к шагу 2.

Изменение матрицы

Шаг 8. Вычислить $\Delta = \min_{\substack{i \in I^+ \\ k \in K^-}} [c'_{ik}]$; положить $c'_{ik} = c'_{ik} - \Delta$, если $i \in I^+$ и $k \in K^-$ и $c'_{ik} = c'_{ik} + \Delta$, если $i \in I^-$ и $k \in K^+$; оставить без изменения в остальных случаях. Сохранить пометки и вернуться к шагу 3.

Шаг 9. Найдено оптимальное решение. Грузы транспортируются только по отмеченным ребрам и, если элемент (i, k) отмечен, c'_{ik} дает величину груза, транспортируемого по ребру $(x_i,$

Следует заметить, что в вышеприведенном алгоритме матрица C' служит двум целям. Неотмеченные элементы дают предварительные стоимости. (Некоторые из этих элементов могут быть нулями.) Все же отмеченные элементы имеют нулевые стоимости, но места, где они расположены, используются на самом деле для хранения величин грузов, транспортируемых по соответствующим ребрам. Это избавляет от необходимости использования отдельной матрицы для хранения транспортируемых величин.

5.2. Пример

Решим ТЗ с нижеприводимой матрицей стоимостей, где векторы, расположенные слева от матрицы и над ней, дают величины предложения α_i и спроса β_k соответственно источников и стоков. (Векторы весов $[\pi_i]$ и $[\pi_k]$ здесь не нужны.)

	β_k	2	3	3	5	2
α_i		1	2	3	4	5
5	1	2	0	1	1	3
4	2	0	4	3	0	0
6	3	3	1	0	0	1

Вышеприведенная матрица редуцируется к следующей:

	β_k	2	3	3	5	2
α_i		1	2	3	4	5
5	1	5	3	4	5	6
4	2	2	6	5	3	2
6	3	6	4	3	4	4

После нескольких применений шагов 2, 3, 6 и 7 с непосредственным обнаружением аугментальной цепи матрица C', векторы

α_i и β_k и пометки будут такими:

		β_k	0	0	0	0	2	
α_i			1	2	3	4	5	
2	1		2	3*	1	1	3	0
0	2		2*	4	3	2*	0	
0	3		3	1	3*	3*	1	
								p_i
				1				p_k

Вычисленное Δ равно 1, а новая матрица C′ такова:

		β_k	0	0	0	0	2	
α_i			1	2	3	4	5	
2	1		1	3*	0	[0]	2	0
0	2		2*	5	3	[2*] ←	[0]	4
0	3		3	2	3*	3*	1	3
								p_i
				1	1	1	2	p_k

Пометки делаются в следующем порядке: $p_{\text{столб }3} = p_{\text{столб }4} = 1$, $p_{\text{стр }3} = 3$ (или 4), $p_{\text{стр }2} = 4$, $p_{\text{столб }5} = 2$, после чего помечен столбец 5 с $\beta_5 = 2$ и найдена аугментальная цепь (показанная выше). Вычисленное ε равно 2, и после шага 7 матрица такова:

		β_k	0	0	0	0	0
α_i			1	2	3	4	5
0	1		1	3*	0	2*	2
0	2		2*	5	3	0	2*
0	3		3	2	3*	3*	1

На шаге 2 алгоритм заканчивает свою работу, так как все $\alpha_i = 0$ и решение в вышеприведенной матрице показано отмеченными элементами. Для этой ТЗ минимальная стоимость равна $3 \times 3 + 2 \times 5 + 2 \times 2 + 2 \times 2 + 3 \times 3 + 3 \times 4 = 48$.

6. Задача о покрытии

Задача о покрытии наименьшей мощности (ЗПНМ) была сформулирована во введении, и она состоит в нахождении такого множества E ребер графа $G = (X, A)$, что число ребер в E минимально и каждая вершина из G является концевой вершиной по крайней мере одного ребра из E. Теперь, основываясь на следующей теореме, мы покажем, что ЗПНМ и ЗНПС эквивалентны.

Теорема 6. *Если E^* — наименьшее покрытие и если для каждой вершины x_i со степенью $d_i^{E^*} > 1$ удалить все ребра, инцидентные x_i, за исключением одного, то оставшееся множество ребер M^* будет наибольшим паросочетанием.*

Доказательство. Так как в вышеприведенной конструкции $d_i^{M^*} \leqslant 1$, то никакие два ребра в M^* не будут смежными и M^* является паросочетанием. Поэтому достаточно показать, что M^* — наибольшее паросочетание. Допустим, что $|M^*|$ не максимально. Тогда по теореме 1 существует некоторая аугментальная цепь из какой-нибудь экспонированной вершины x_i (т. е. вершины с $d_i^{M^*} = 0$) к другой экспонированной вершине x_k. Так как $d_i^{M^*} = d_k^{M^*} = 0$, то отсюда вытекает, что по крайней мере одно ребро, ведущее от x_i к другой вершине ($\neq x_k$), можно удалить, и то же относится к вершине x_k. Пусть одно из удаленных ребер, инцидентных x_i, будет a_1, а одно из удаленных ребер, инцидентных x_k, будет a_2. Пусть P — аугментальная цепь (с $2p + 1$ ребрами) от x_i к x_k. Если теперь удалить из E^* ребра из M^*, принадлежащие цепи P, и два ребра a_1 и a_2, то оставшееся множество (скажем, $\bar{E}$) все еще будет покрытием, так как каждая вершина из P будет концевой для некоторого ребра из множества E^*. Но $\bar{E}$ получено удалением из E^* $p + 2$ ребер и добавлением $p + 1$ ребер; следовательно, $|\bar{E}| < |E^*|$. Это противоречит предположению о том, что E^* — наименьшее покрытие. Значит, M^*, построенное в соответствии с теоремой, является наибольшим паросочетанием.

Следующая теорема аналогична теореме 6 и точно так же доказывается [30].

Теорема 7. *Если M^* — наибольшее паросочетание графа G и для каждой экспонированной вершины x_i добавить ребро,*

инцидентное x_i, то получившееся множество ребер E^* будет наименьшим покрытием графа G.

Общая ЗПО, когда каждому ребру приписана некоторая стоимость, здесь обсуждаться не будет. Она может быть решена с помощью алгоритма, очень похожего на алгоритм для ЗМП. Подробный алгоритм для ЗПО дал Уайт [30].

7. Задачи

1. Найти наибольшее паросочетание графа, изображенного на рис. 12.26. Построить затем наименьшее покрытие этого графа.

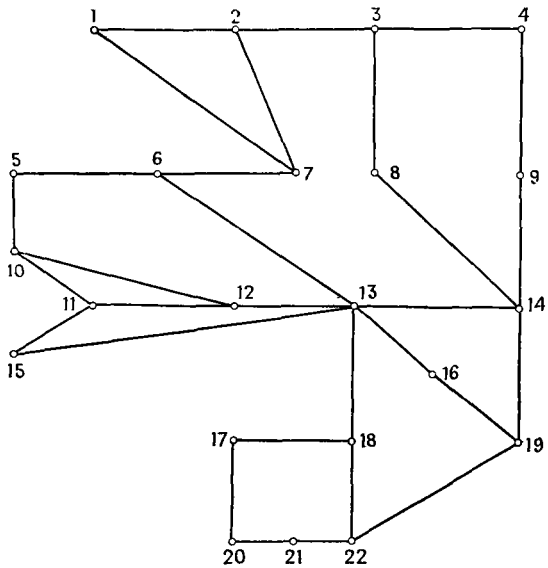


Рис. 12.26.

2. Найти максимальное паросочетание графа, изображенного на рис. 12.27, где числа у ребер являются стоимостями (весами).

3. Показать, что в двудольном графе $G = (X^a \cup X^b, A)$ совершенное паросочетание существует тогда и только тогда, когда для каждого $S \subset X$, $|S| \leqslant |\Gamma(S)|$. (Этот результат известен как теорема Кёнига и Холла [5].)

4. Для графа, изображенного на рис. 12.28, найти какой-либо остовный подграф G_p, степени вершин которого задаются вектором $[\delta_i] = [1, 2, 1, 2, 3, 1, 3, 2]$.

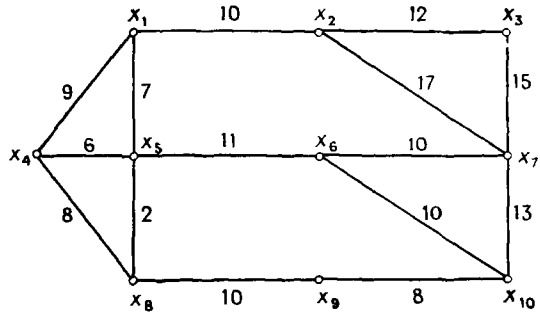


Рис. 12.27.

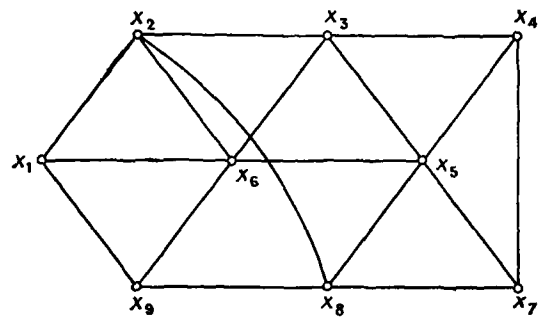


Рис. 12.28.

5. Показать, что в матрице, имеющей несколько нулевых и несколько ненулевых элементов, минимальное число линий (т. е. строк и столбцов), содержащих все нули, равно максимальному числу нулей, расположенных в разных строках и разных столбцах. (Этот результат известен как теорема Кёнига.)

6. Найти решение ЗН с минимальной стоимостью, где матрица $\mathbf{C}$ равна

$$\mathbf{C} = \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 17 & 8 & 12 & 9 & 14 & 6 \\ 2 & 15 & 13 & 18 & 15 & 10 & 4 \\ 3 & 14 & 16 & 8 & 12 & 5 & 9 \\ 4 & 18 & 7 & 14 & 9 & 11 & 13 \\ 5 & 7 & 16 & 11 & 14 & 6 & 10 \\ 6 & 18 & 9 & 13 & 5 & 15 & 7 \end{array}$$

7. Определим минимаксную ЗН как задачу нахождения такого назначения с матрицей **C**, чтобы максимальная стоимость элемента этого назначения была минимальной. Решить минимаксную ЗН для матрицы **C** из задачи 6.

8. Решить нижеприведенную транспортную задачу, где (4×5)-матрица является матрицей стоимостей **C**, вектор-столбец является вектором снабжения (предложений), а вектор-строка — вектором спроса (потребностей).

3	4	2	7	6

9	17	18	12	9	14
6	15	13	18	15	10
5	14	16	8	12	5
7	18	7	14	9	11

(*Замечание.* Свести задачу к эквивалентной задаче, в которой сумма потребностей равна сумме предложений.)

9. Рассмотрим транспортную задачу с матрицей стоимостей **C**, приведенной ниже, и векторами снабжения и потребностей

$$[\alpha_i]^1 = [10, 15, 20]$$

$$[\beta_k]^1 = [5, 12, 13, 15]$$

$$\mathbf{C} = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 1 & 25 & 10 & 2 & 30 \\ 2 & 5 & 15 & 20 & 10 \\ 3 & 100 & 65 & 0 & 2 \end{array}$$

Оптимальное решение этой задачи дается перевозками, показанными в верхних левых углах следующей матрицы:

$$[\xi_{ik}] = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 1 & & 2/3 & 8/7 & \\ 2 & 5/6 & 10/9 & & \\ 3 & & & 5/6 & 15/15 \end{array}$$

Стоимость этого решения равна 241, а величина транспортируемых грузов равна 45 единицам. Рассмотрим новую задачу с той же самой матрицей стоимостей, но с векторами снабжения и потребностей

$$[\alpha_i]^2 = [10, 15, 21],$$
$$[\beta_k]^2 = [6, 12, 13, 15].$$

Оптимальное решение дается перевозками, показанными в нижних правых углах вышеприведенной матрицы $[\xi_{ik}]$. Стоимость этого решения равна 239, а величина транспортируемых грузов — 46 единицам.

Объяснить, почему для данной матрицы стоимостей можно транспортировать большую величину грузов при меньшей полной стоимости, и особенно обратить внимание на то, что дуги, вдоль которых производится перевозка, остаются прежними (см. [28]).

8. Список литературы

1. Balinski M. L. (1965), Integer programming; Methods, uses, computations, *Man. Sci.*, 12, p. 253.
2. Balinski M. L. (1969), Labelling to obtain a maximum matching, Combinatorial Math. and its Appl., Bose and Dowling, Eds., Univ. of North Carolina Press, p. 585.
3. Balinski M. L. (1970), On maximum matching, minimum covering and their connections, Proc. Princeton Symp. on Math. Programming, Kahn, Ed., Princeton Univ. Press, p. 303.
4. Balinski M. L., Gomory R. E. (1964), A primal method for the assignment and transportation problems, *Man. Sci.*, 10, p. 578.
5. Берж К. (1962), Теория графов и ее применения, *ИЛ*, М.
6. Bourgeois F., Lassalle J. C. (1971), Algorithm 415 — Algorithm for the assignment problem, *Comm. of ACM*, 14, p. 805.
7. Bourgeois F., Lassalle J. C. (1971), An extension of the Munkres algorithm for the assignment problem to rectangular matrices, *Comm. of ACM*, 14, p. 802.
8. Dantzig G. B. (1963), Linear programming and extensions, Princeton Univ. Press, Princeton, New Jersey.
9. Desler J. F. Hakimi S. L. (1969), A graph-theoretic approach to a class of integer programming problems, *Ops. Res.*, 17, p. 1017.
10. Edmonds J. (1965), Paths, trees and flowers, *Canadian Math. J.*, 17, p. 449.
11. Edmonds J. (1965), Maximum matching and a polyhedron with 0-1 vertices, *J. of Res., Nat. Bur. of Stand.*, **69B**, p. 125.
12. Edmonds J. (1967), An introduction to matching, Summer seminar on mathematics of the decision sciences, Stanford University.
13. Edmonds J., Johnson E. L. (1970), Matching: A well solved class of integer linear programs, Proc. Calgary Intl. Conf. on Combinatorial Structures and their Appl., Gordon and Breach, New York, p. 89.
14. Edmonds J. (1962), Covers and packings in a family of sets, *Bull. Amer. Math. Soc.*, 68, p. 494.
15. Egervary E. (1953), On combinatorial properties of matrices, translated by H. W. Kuhn, Office of Naval Research Logistics Project Report, Dept. Math., Princeton University.

16. Форд Л., Фалкерсон Д. (1966), Потоки в сетях, «Мир», М.
17. Garfinkel R. S., Nemhauser G. L. (1972), Integer Programming, Wiley, New York.
18. Glover F. (1967), Maximum matching in a convex bipartite graph, *Nav. Res. Log. Quart.*, **14**, p. 313.
19. Gordon B. E. (1971), The maximum matching problem — A comparison of the Edmonds and Balinski algorithms, Graduate School of Management, University of Rochester.
20. Ivanescu P. L., Rudeanu S. (1968), A pseudo-boolean approach to matching problems in graphs with applications to assignment and transportation problems, Théorie des Graphes, Dunod, Paris.
21. Klein M. (1967), A primal method for minimal cost flows with applications to the assignment and transportation problems, *Man. Sci.*, **14**, p. 205.
22. Konig D. (1950), Theorie der endlichen und unendlichen Graphen, Chelsea, New York.
23. Kuhn H. W. (1955), The Hungarian method for the assignment problem, *Nav. Res. Log. Quart.*, **2**, p. 83.
24. Kuhn H. W. (1956), Variants of the Hungarian method for the assignment problem, *Nav. Res. Log. Quart.*, **3**, p. 253.
25. Morrison D. R. (1969), Matching algorithms, *J. of Combinatorial Theory*, **6**, p. 20.
26. Munkres J. (1957), Algorithms for the assignment and transportation problems, *J. of SIAM (Appl. Math.)*, **5**, p. 32.
27. Norman R. Z., Rabin M. O. (1959), An algorithm for a minimum cover of a graph, *Proc., Amer. Math. Soc.*, **10**, 315.
28. Szwarc W. (1970), The transportation paradox, Report 199, Carnegie — Mellon University, Pittsburg, Pennsylvania.
29. Thacker B. G. (1972), Matchings in weighted graphs, M. Sc. Thesis, Imperial College, London University.
30. White L. J. (1967), A parametric study of matchings and covering in weighted graphs, Ph. D. Thesis, University of Michigan.
31. Witzgall C., Zahn C. T., Jr. (1965), Modification of Edmonds' maximum matching algorithm, *J. of Res. Nat. Bur. of Stands*, **69B**, p. 91.
32. Yaspan A. (1966), On finding a maximal assignment, *Ops. Res.*, **14**, p. 646.

Приложение 1

МЕТОДЫ ПОИСКА, ИСПОЛЬЗУЮЩИЕ ДЕРЕВО РЕШЕНИЙ

1. Принцип поиска, использующий дерево решений

Основной принцип, на котором базируются методы поиска с деревом решений, состоит в разбиении начальной задачи P_o на некоторое число подзадач $P_1, P_2, \ldots, P_k$ (в целом представляющих всю задачу P_o) с последующей попыткой *разрешить* каждую из этих подзадач. Выражение *разрешить* мы понимаем так:

либо (i) найти оптимальное решение,

либо (ii) показать, что значение оптимального решения хуже, чем для полученного до этого наилучшего решения,

либо (iii) показать. что подзадача не является допустимой.

Это разбиение описывается *деревом* на рис. П.1, причем вершины изображают подзадачи.

Смысл разбиения задачи P_o на некоторое число подзадач состоит в том, что или эти подзадачи проще разрешить, или они имеют меньший размер, или обладают структурой, не присущей первоначальной задаче P_o. Но, вообще говоря, все еще может оказаться, что подзадачу P_i нельзя разрешить, и эта подзадача сама разбивается на новые подзадачи $P_{i_1}, P_{i_2}, \ldots, P_{i_r}$, как это показано на рис. П.2. Это разбиение (называемое также *ветвлением*), повторяется для каждой подзадачи, которая не может быть разрешена.

На любом этапе полное множество подзадач, требующих разрешения, представляется множеством концевых вершин (т. е. вершин степени 1) всех цепей, исходящих из корня дерева решений. (Корень этого дерева изображает начальную задачу P_o.) Эти вершины называются *висячими* вершинами, и на рис. П.2 это $P_1, \ldots, P_{i-1}, P_{i_1}, \ldots, P_{i_r}, P_{i+1}, \ldots, P_k$.

Если поиск исчерпан, то очевидно, что множество подзадач, на которые разбита задача, должно представлять всю задачу. Таким образом, если задача P_i разбита на r подзадач $P_{i_1}, \ldots, P_{i_r}$, то

$$\{P_{i_1}\} \cup \{P_{i_2}\} \cup \ldots \cup \{P_{i_r}\} = \{P_i\}, \qquad (\text{П.1})$$

где $\{P\}$ обозначает множество всех допустимых решений задачи P.

Так как соотношение (П.1) должно быть применено к каждому разбиению, то

$$\{P_o\} = \cup \{P(j) \mid P(j) — \text{висячая вершина дерева}\}. \qquad (П.2)$$

В случаях когда требуется перебрать все решения задачи P_o (а не только найти оптимальное решение), желательно уметь

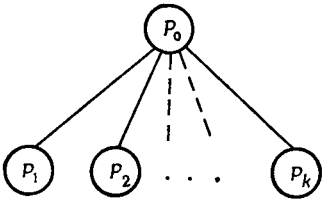


Рис. П.1. Разбиение задачи P_0 на подзадачи.

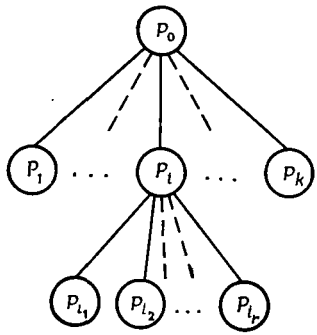


Рис. П.2. Дерево после ветвления в вершине P_i.

перебирать решения с помощью вышеприведенного разбиения задачи на подзадачи и перебирать решения каждой из этих подзадач. В этом случае нужно избежать дублирования построенных решений, т. е. нужно разбивать задачу P_i на подзадачи $P_{i_1}, \ldots, P_{i_r}$ так, чтобы

$$\{P_{i_s}\} \cap \{P_{i_q}\} = \varnothing \qquad (П.3)$$

для любых двух подзадач P_{i_s} и P_{i_q}, для которых $s \neq q$.

Соотношение (П.3) определяет собственное разбиение задачи P_i. Хотя условие (П.3) не является необходимым для полноценного поиска с деревом решений, оно тем не менее имеет большие выгоды с вычислительной точки зрения, так как

(а) для задачи оптимизации P_o оптимальное решение является решением одной и только одной подзадачи, представляемой висячей вершиной;

(б) для задачи полного перебора объединение множеств решений подзадач, представляемых висячими вершинами, дает множество всех решений задачи P_o без дублирования.

2. Некоторые примеры ветвления

Рассмотрим задачу P_i с n переменными, в которой некоторая переменная ξ может принимать только четыре возможных значения, скажем, a, b, c и d.

(а) Возможно разбиение P_i на четыре подзадачи P_{i_1}, P_{i_2}, P_{i_3} и P_{i_4}, причем для подзадачи P_{i_1} мы полагаем $\xi = a$, для P_{i_2} полагаем $\xi = b$, для P_{i_3} — $\xi = c$ и для P_{i_4} — $\xi = d$. Каждая

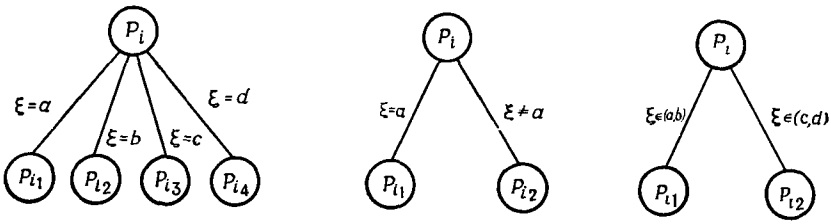


Рис. П.3. Три возможных способа ветвления в вершине P_i.

из подзадач $P_{i_1}, \ldots, P_{i_4}$ содержит $n - 1$ переменных и, следовательно, допускает более простое решение, чем задача P_i (рис. П.3 (а)).

(б) Возможно другое разбиение P_i на две подзадачи P_{i_1} и P_{i_2}, где для P_{i_1} мы полагаем $\xi = a$, а для P_{i_2} полагаем $\xi \neq a$, т. е. ξ равно b, c или d (рис. П.3 (б)).

(в) Еще одно возможное разбиение P_i на две подзадачи P_{i_1} и P_{i_2}, где для P_{i_1} $\xi = a$ или b, а для P_{i_2} $\xi = c$ или d (рис. П.3 (в)).

Все три ветвления являются допустимыми и удовлетворяют условию (П.3). Какому из них отдать предпочтение — зависит от природы решаемой задачи, причем возможности типа (а) или (б) используются чаще остальных.

3. Типы поиска, использующего дерево решений

Из вышесказанного видно, что любая подзадача, представляемая висячей вершиной и не поддающаяся разрешению, может быть в любой момент разбита на меньшие подзадачи. Но существует только два основных типа поиска в зависимости от того, как выбирается следующая висячая вершина для продолжения процесса ветвления.

3.1. Поиск по глубине

При этом типе поиска ветвление осуществляется в последней полученной подзадаче до тех пор, пока не будет порождена подзадача, которую можно разрешить. В этом месте делается шаг *возвращения*, т. е. берется предпоследняя порожденная подзадача и ветвление продолжается в соответствующей вершине. При этом типе поиска задачи, получаемые на каждом этапе, хранятся в *стеке* вместе с самой верхней задачей, выбранной для исследования (разрешения или разбиения). Вновь получаемые задачи помещаются в верх этого стека, а когда подзадача разрешена, она уда-

Рис. П.4а. Дерево поиска по глубине.

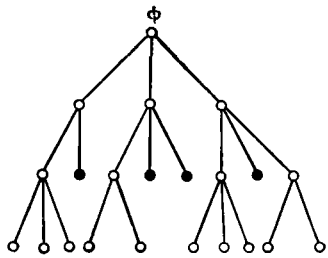


Рис. П.4б. Дерево поиска по ширине.
● Задача решена.

ляется от стека. Вид дерева решений при этом типе поиска, когда разрешается первая подзадача, показан на рис. П.4а, где порядок приоритета исследования получаемых подзадач показан нумерацией.

3.2. Поиск по ширине

При поиске по ширине ветвление происходит от уровня к уровню, так что если на уровне 1 начальная задача P_0 разбивается на подзадачи $P_1, P_2, \ldots, P_k$, то каждая из этих подзадач исследуется раньше, чем задачи уровня 2. Задачи уровня 1, которые не могут быть разрешены, разбиваются на подзадачи уровня 2, и опять все они исследуются *до* исследования какой-либо подзадачи, могущей возникнуть на уровне 3, и т. д. Вид дерева решений при этом типе поиска показан на рис. П.4б.

4. Применение границ

Если задача P_0 подлежит решению как задача оптимизации, то безотносительно к типу поиска этот поиск завершается только тогда, когда будут разрешены все подзадачи, представляемые висячими вершинами. Для ускорения процесса разрешения для каждой из висячих вершин вычисляются нижние или верхние границы (для случаев минимизации или максимизации соответственно). Эти границы дают наименьшее (или наибольшее) возможное значение оптимального решения той подзадачи, которая

соответствует рассматриваемой висячей вершине. Таким образом (для задачи минимизации), если окажется, что нижняя граница для вершины, соответствующей задаче P_i, больше, чем величина наилучшего ответа, полученного ранее при поиске, то в P_i нет необходимости производить дальнейшее ветвление, так как в $\{P_i\}$ нет решения, лучшего, чем текущий наилучший ответ. В соответствии с толкованием (II) термина «разрешение задачи» из разд. 1 подзадача P_i окажется автоматически разрешенной.

5. Функции ветвления

Как при поиске по глубине, так и при поиске по ширине выбор очередной вершины для ветвления не был полностью определен.

При поиске по глубине, когда после ветвления задача P_i разбивается на подзадачи $P_{i_1}, \ldots, P_{i_r}$, очередное ветвление, как уже говорилось, производится в одной из этих только что порожденных подзадач. Но мы не указали, в какой именно, и любая из них может рассматриваться как «последняя порожденная». При поиске по ширине, как уже было сказано, все подзадачи данного уровня должны исследоваться *до* исследования задач следующего уровня, но не был указан порядок их исследования.

Функция ветвления — это функция, которая позволяет «вычислить», какая из *допустимых* вершин должна использоваться при следующем ветвлении. Для вершины, соответствующей подзадаче P_j, эта функция является некоторой мерой вероятности того, что оптимальное решение всей задачи P_0 является решением для P_j. Совершенно очевидно, что вершина, соответствующая подзадаче с бóльшими шансами на оптимальное решение, должна пользоваться правом преимущественного выбора при очередном ветвлении. Можно указать несколько эвристических мер этой вероятности, причем одна из полезных мер связана просто с вычислением для вершин нижних или верхних границ. Для такой меры вершина с более низкой нижней границей (для случая минимизации) считается имеющей большую вероятность.

После введения понятия функции ветвления сразу же возникает мысль о другом типе поиска с деревом решений (в дополнение к описанным ранее поискам по глубине и ширине). Можно использовать функцию ветвления и так, чтобы она *полностью* определяла выбор следующей для ветвления вершины. Например, если значениями функции ветвления являются границы вершин (нижние и верхние), как упоминалось выше, то всегда можно производить ветвление в той висячей вершине, нижняя граница которой наименьшая. Этот тип поиска является, вообще говоря, гибридом поисков по глубине и ширине, хотя в литературе он часто называется поиском по ширине.

Предметный указатель

Активный цикл *см.* Маршрут замкнутый активный
Алгоритм «беспорядка» [out-of-kilt»] 339
— венгерский 405
— — для задачи о назначениях 405—407
— — — транспортной задачи 413
— двойственный решения задачи о потоке минимальной стоимости 351—353
— *Дейкстры* решения задачи о кратчайшем пути между двумя заданными вершинами s и t с неотрицательной матрицей весов 174—183
— *Краскала* построения кратчайшего остова графа 160—162
— направленного древовидного поиска для задачи о p-медиане 138
— — поиска для задачи о p-медиане 139—144
— нахождения абсолютного p-центра 114, 115
— основной для задачи о потоке минимальной стоимости 339—342
— поиска, использующего дерево решений для задачи о коммивояжере 285—295
— приближенный для задачи о p-медиане 139—141
— *Прима* построения кратчайшего остова графа 162—163
— расстановки пометок в задаче о максимальном потоке 314—315
— решения задачи китайского почтальона 331, 332
— — — о K кратчайших путях между двумя заданными вершинами 195, 196
— — — — кратчайшем пути между двумя заданными вершинами s и t с общей матрицей весов 183—189
— — — — назначения 405
— — — — — матричная форма 406, 407
— — — — наибольшем паросочетании (ЗНПС) 381—383
— — — — наименьшем покрытии (ЗНП), использующий дерево поиска 55
— — — — покрытии наименьшей мощности (ЗНПМ) 416
— — — — — разбиении (ЗНР) 55
— — — — потоке между каждой парой вершин 331—334
— — — — раскраске и использованием дерева поиска 88—90
— *Робертса и Флореса* порождения гамильтонова цикла 249—253
— *Флёри* построения эйлерова цикла 230
— *Флойда* решения задачи о кратчайшем пути между всеми парами вершин 189—190
— *Хакими* нахождения абсолютного центра 104, 105
— — модифицированный 107—110
— штрафования вершин для задачи коммивояжера 285—295
Алгоритмы приближенные решения задачи о раскраске 90—91
Антибаза [contrabasis] 38

База [basis] 37—40
— сильная [power] 39
Булевское (логическое) выражение 64

Вершина [vertex] 11
— внешняя [outer] 374—378
— внутренняя [inner] 374—375
— конечная [final] 11
— концевая [terminal] 11
— начальная [initial] 11
Вершина несущественная, избыточная [inessential] 33
— пропускная способность 326
— существенная, неотъемлемая [essential] 33
— экспонированная [exposed] 371
Внешне устойчивое множество *см.* Доминирующее множество вершин
Внутреннее произведение вершин 245
Выбор места для склада 129
— проекта 45

Гипотеза четырех красок 79
Граф [graph] 11
— антисимметрический 20
— взвешенный 15
— двудольный [bipartite] 21, 405, 412
— — неориентированный 21
— — ориентированный 21
— дополнение 46
— инкрементальный [incremental] 321, 339, 350, 352, 360
— *Куратовского* 23
— *Муна-Мозера* 70
— неориентированный [nondirected] 11
— — двойник 11
— непланарный [nonplanar] 23
— несвязный [disconnected] 23
— односторонне связный или односторонний [unilateral] 23
— ориентированный [directed] 11
— остовов [tree graph] 157
— планарный [planar] 22, 79
— полный [complete] 19
— — антисимметрический 20
— — симметрический 20
— реберный [line] 237
— редуцированный [reduced] 254
— r-хроматический [r-chromatic] 75
— сильно связный или сильный [strong] 23
— симметрический [simmetric] 20
— слабо связный или слабый [weak] 23
— со взвешенными вершинами [vertex-weighted] 15
— — — дугами [arc-weighted] 15
— транзитивный [transitive] 33
— унитарный [unitary] 323
Густота *см.* Кликовое число

Дерево [tree] 145—172
— альтернирующее 374
— аугментальное [augmenting] 375
— венгерское 380—382
— ориентированное [directed] 145—147, 240
— остовное [spanning tree] (см. Остов) 145—224
— — длиннейшее 163
— — процедура порождения 149—157
— — расщепление [division] 153
— решения для поиска [search tree] 422—427
— — ветвление 422, 424
— — границы 426
— — для поиска по глубине 425
— — — — — ширине 425
— — висячая вершина 423
— — разбиение 424
— — функции ветвления 426—427
— сращивание [merging] 153
— цветущее [blossomed] 376
— *Штейнера* наикратчайшее 167—169
— элементарное преобразование 149
Диаметр графа 11, 125

Доминирующее множество вершин 40, 40
— — — минимальное [minimal] 50
— — — наименьшее [minimum] 50
Достижимое множество 29
Достижимость [reachability] 23
Дуга [arc] 11
— вес [weight] (длина [length], стоимость или цена [cost] 15, 201
— надежность [reliability] 201
— нижняя граница потока через 310
— обратная 313, 356
— поток, входящий в 354
— — выходящий из 354
— пропускная способность 202, 282, 313
— прямая 313, 356

Задача государственного районирования [political districting] 65
— об остовном графе с предписанными степенями [degree-constrained partial graph problem] 368, 411—414
— китайского почтальона 231—237
— нахождения допустимого потока минимальной стоимости 310
— о доставке молока или почты [delivery of post] 231
— — Кенигсбергских мостах 228
— — коммивояжере 242—309
— — — минимаксная 244
— — — минисуммная 244
— — — нижняя граница из задачи о кратчайшем остове 266, 279
— — — — — — — — — назначениях 265, 297—303
— — *K* кратчайших путях между двумя заданными вершинами 195
— — кратчайшем пути между заданными вершинами *s* и *t* 175—189
— — — — — — — — — — — с неотрицательной матрицей весов 177—183
— — *K* ферзях на шахматной доске 70
— — максимальном паросочетании (ЗМП) 368—371
— — — потоке (от *s* к *t*) 310—325
— — минимальном покрытии (ЗНПО) 369
— — многопродуктовом потоке [multicommodity] 311, 325
— — назначениях (ЗН) [assignment problem] 284, 404—411
— — наибольшем паросочетании (ЗНПС) 370, 375
— — наименьшем покрытии 43, 53—68
— — — — вычисление нижней границы 60
— — — — приложения 63—68
— — — — упрощение 54
— — покрытии наименьшей мощности (ЗПНМ) [minimum cardinality covering problem] 370, 416
— — потоке минимальной стоимости от *s* к *t* 310, 339
— — потоках с выигрышами 311, 353—364
— — раскраске 375
— — — решение методом динамического программирования 80—84
— — — — — 0-1 программирования 84—46
— — — свед'ние к ЗНП 86—88
— — распределении ресурсов 94
— размещения минимаксная 98, 127
— — минисуммная 98, 127
— сетевого планирования [network planning] 65
— синхронизации линии сборки [assembly line balancing] 65
— теории расписаний 94
— транспортная (ТЗ) 371, 412—416
Задача *Штейнера* 166—169
— — евклидова 167
— — линейная 169
— — на графах 166, 167

Информационный поиск [retrieval information] 63
Исследование структуры организации 39
Источник [source] 310

Клика графа [clique] 46
Кликовое число [clique number] 43
Компонента графа односторонняя 24
— — сильная 24, 33—36
— — слабая 24
Компрессия [compression] матрицы 297
Константа «проникновения» [penetration] 114
Контрадостижимое множество [reaching set] 30
Контур *см* Орцепь замкнутая простая 17
— гамильтонов 17, 157, 237, 242—309
— — алгебраический метод нахождения 245—249
— независимый [independent] 218
Корень [root] дерева 374
— ориентированного дерева 146
— — — замена 193
Коцикломатическое число 218
Кратные центры [multiple centres] 111
Кратчайшее дерево *Штейнера* 166—167
Кратчайший остов графа [shortest spanning tree] 158—161
Критический путь [critical path] 197—200

λ-оптимальность 140—141

Максимальный полный подграф *см* Клика 46
Маршрут [chain] 4
— аугментальный [augmenting] 356
— — инкрементальная пропускная способность [incremental capacity] 356
— выигрыш 356
— замкнутый [cycle] 17
— — активный [active cycle] 358—364
Маршруты полетов самолетов 64
Матрица достижимостей [reachability matrix] 29, 30, 31
— инциденций [incidence matrix] 26, 148, 155, 170, 226, 239, 379
— контрадостижимостей [reaching] 30, 31
— ограниченных достижимостей 33
— редуцированная 406
— смежности [adjacency matrix] 25
— — модифицированная 245
Медиана [median] 98, 127—143
— абсолютная 130
— внешняя 128
— внешне-внутренняя 129
— внутренняя 128
— кратная *см* *p*-медиана 129
Метод критического пути (МКП) [critical path method] 197

Наименьшее доминирующее множество ребер *см* Наименьшее покрытие
Независимое множество вершин [independent vertex set] 44
— — — максимальное [maximal] 44—48, 80, 86, 87
— — — наибольшее [maximum] 45, 65
— — ребер [independent link set] 66
— — — наибольшее 66

Область 114
Обход лабиринта 240

Эрцепь см. Цепь ориентированная 14
— замкнутая 17
— — простая [elementary circuit] 17
Эстов [spanning tree] 145, 224
— число 148
Отображение [mapping] 11

Паросочетание [matching] 66, 368—420
— максимальное [maximal] 389
— наибольшее [maximum] 66, 368
— совершенное [perfect] 389
Передаточное число [transmission number] 127—128
— — внешнее [out] 128
— — внутреннее [in] 128
ПЕРТ 197
Петля [loop] 16
Плотность см Кликовое число
Подграф [partial subgraph] 19
— максимальный [maximal subgraph] 23, 24
— остовный [partial graph] 18
— порожденный [subgraph] 18
Покрытие [covering] 66, 67, 369
— минимальное [minimal] 369
— наименьшее [minimum] 66, 67
Полустепень захода [indegree] 18
— исхода [outdegree] 18
Пометка предшествования 153
— — изменение 153
Поток [flow] 310
— аугментальная цепь [flow-augmenting chain] 313
— в графах с выигрышами 356
— — — — — — в вершинах 364
— — — — многими источниками и стоками 325
— — — — пропускными способностями дуг и вершин 326
— допустимый [feasible flow] 310, 353—358
— — максимальный 354
— — оптимально-максимальный 354
— — оптимальный 354
— конформальный [conformal] 325
Потоковая эквивалентность 330
Проверка электрических, телефонных или железнодорожных линий 231
Псевдовершина [pseudovertex] 375
Пустое множество 11
Путь [path] 13
— вес, длина или стоимость [length] 15
— длина или мощность [cardinality] 16
— замкнутый [path] 16
— — элементарный 17
— кратчайший 173, 189—193
— надежность 201, 202
— ответвление [deviation] 195
— пропускная способность 202
— самый длинный 198
— с наибольшей приведенной пропускной способностью 206—211
p-кратный внешний центр [*p*-outcentre] 112
— внутренний центр [*p*-incentre] 112
p-медиана 129
— абсолютная 130
— внешняя 130
— внутренняя 130
— обобщенная 132, 139
p-центр (кратный центр) 41, 111, 112
— абсолютный 112, 113
— — нахождение 113—123

Радиус 101
— абсолютный внешний 103
— — внутренний 101
— внешне-внутренний 102
— внешний 101
Радиус внутренний 101
Разделение [separation] 99—101
— внешнее [out] 100
— внутреннее [in] 100
Размещение [location] 98
— аварийных служб и пунктов обслуживания 101—102, 106, 107
— нескольких центров обслуживания 112
— центров 51, 52, 98—125
Разрез [cut-set] 221—225, 312, 313
— величина 312
— для ориентированного графа 223, 224
— правильный [proper] 222, 223
— фундаментальный 224, 225
— — матрица 225, 226, 239
Раскраска [colouring] 75—96
— оптимальная независимая 80, 84
Ребро [link] 11
— искусственное 232
r-подграф 80
— максимальный 80—84

Смежные дуги 14
— вершины 14
Соответствие 11
— обратное 13
Специальный остовный подграф [equally partial] 391
Степень вершины [degree] 18
— — *k*-шаговая 91
Строгое пересечение (SI) [strict intersection] 117—118
Сток [sink] 310
(*s-t*)-разрез 202-206

Теорема *Кенига* 418
— — и *Холла* 417
— о максимальном потоке и минимальном разрезе 312, 313
— — пяти красках 79
Точка *Штейнера* 168, 169
Транзитивное замыкание графа 33
Турнир [tournament] 21

Хроматическое число [chromatic number] 75
— — верхняя оценка 78
— — нижняя оценка 77, 78

Цветок [blossom] 375—379
— крайний [outermost] 375
— срезание [shrinking] 375—379
Центр графа 98-103
Цепь альтернирующая [alternating path] 371
— аугментальная [augmenting path] 372, 372
— ориентированная (орцепь) [simple path] 14
— простая [elementary path] 14
— эйлерова *см* Эйлеров цикл
Цикл гамильтонов 17, 242—309
— ориентированный (орцикл) 17
— — матрица 225, 239
— — мультицепной метод нахождения 253—259
— — сравнение методов поиска 259—262
— фундаментальный 220, 221
— эйлеров 227—240
Цикломатическое число [cyclomatic number] 217, 218

Число Бетти *см* Цикломатическое число
— внешнего разделения 100
— внутреннего разделения 100
— доминирования 43
— независимости [independence number] 43, 44

Оглавление

Предисловие редактора перевода 5
Предисловие 7

Глава 1. Введение 11

1. Графы. Определение 11
2. Пути и маршруты 13
3. Петли, ориентированные циклы и циклы 16
4. Степени вершины 18
5. Подграфы 18
6. Типы графов 19
7. Сильно связные графы и компоненты графа 23
8. Матричные представления 25
9. Задачи 27
10. Список литературы 28

Глава 2. Достижимость и связность 29

1. Введение 29
2. Матрица достижимостей и контрадостижимостей 29
3. Нахождение сильных компонент 33
4. Базы 37
5. Задачи, связанные с ограниченной достижимостью 40
6. Задачи 41
7. Список литературы 42

Глава 3. Независимые и доминирующие множества. Задача о покрывающих множествах 43

1. Введение 43
2. Независимые множества 44
3. Доминирующие множества 50
4. Задача о наименьшем покрытии 53
5. Приложения задачи о покрытии 63
6. Задачи 68
7. Список литературы 71

Глава 4. Раскраски 75

1. Введение 75
2. Некоторые теоремы и оценки, относящиеся к хроматическим числам 76
3. Точные алгоритмы раскраски 79
4. Приближенные алгоритмы раскрашивания 90
5. Обобщения и приложения 92
6. Задачи 94
7. Список литературы 97

Глава 5. Размещение центров 98

1. Введение 98
2. Разделения 99
3. Центр и радиус 101
4. Абсолютный центр 102
5. Алгоритмы нахождения абсолютных центров 104
6. Кратные центры (p-центры) 111
7. Абсолютные p-центры 112
8. Алгоритм нахождения абсолютных p-центров 114
9. Задачи 123
10. Список литературы 126

Глава 6. Размещение медиан в графе 127

1. Введение 127
2. Медиана графа 127
3. Кратные медианы (p-медианы) графа 129
4. Обобщенная p-медиана графа 132
5. Методы решения задачи о p-медиане 133
6. Задачи 141
7. Список литературы 143

Глава 7. Деревья 145

1. Введение 145
2. Построение всех остовных деревьев графа 148
3. Кратчайший остов (SST) графа 158
4. Задача Штейнера 166
5. Задачи 168
6. Список литературы 172

Глава 8. Кратчайшие пути 175

1. Введение 175
2. Кратчайший путь между двумя заданными вершинами s и t 177
3. Кратчайшие пути между всеми парами вершин 189
4. Обнаружение циклов отрицательного веса 191
5. Нахождение K кратчайших путей между двумя заданными вершинами 198
6. Кратчайший путь между двумя заданными вершинами в ориентированном ациклическом графе 197
7. Задачи, близкие к задаче о кратчайшем пути 201
8. Задачи 211
9. Список литературы 214

Глава 9. Циклы, разрезы и задача Эйлера 217

1. Введение 217
2. Цикломатическое число и фундаментальные циклы . . . 217
3. Разрезы 221
4. Матрицы циклов и разрезов 225
5. Эйлеровы циклы и задача китайского почтальона 227
6. Задачи 239
7. Список литературы 241

Глава 10. Гамильтоновы циклы, цепи и задача коммивояжера 242

1. Введение 242

ЧАСТЬ I

2. Гамильтоновы циклы в графе 245
3. Сравнение методов поиска гамильтоновых циклов 259
4. Простая задача планирования 262

ЧАСТЬ II

5. Задача коммивояжера 265
6. Задача коммивояжера и задача о кратчайшем остове . . . 268
7. Задача коммивояжера и задача о назначениях 284
8. Задачи 304
9. Список литературы 307
10. Приложение 308

Глава 11. Потоки в сетях 310

1. Введение 310
2. Основная задача о максимальном потоке (от s к t) 311
3. Простые варианты задачи о максимальном потоке (от s к t) 325
4. Максимальный поток между каждой парой вершин . . . 329
5. Поток минимальной стоимости от s к t 339
6. Потоки в графах с выигрышами 353
7. Задачи 364
8. Список литературы 367

Глава 12. Паросочетания, транспортная задача и задача о назначениях 368

1. Введение 368
2. Наибольшие паросочетания 371
3. Максимальные паросочетания 389
4. Задача о назначениях 404
5. Общая задача построения остовного подграфа с предписанными степенями 411
6. Задача о покрытии 416
7. Задачи 417
8. Список литературы 420

Приложение 1. Методы поиска, использующие дерево решений . . . 422

1. Принцип поиска, использующий дерево решений 422
2. Некоторые примеры ветвления 424
3. Типы поиска, использующего дерево решений 424
4. Применение границ 426
5. Функции ветвления 426

Предметный указатель 427

Н. Кристофидес
ТЕОРИЯ ГРАФОВ
АЛГОРИТМИЧЕСКИЙ ПОДХОД

Редактор А. Бряндинская. Младший научн. редактор Л. Суркова
Художник О. Камаев. Художественный редактор В. Шаповалов
Технический редактор Н. Иовлева. Корректор А. Рыбальченко

ИБ № 1030

Сдано в набор 24.5.78.
Подписано к печати 2.11.78.
Формат $60\times90^1/_{16}$. Бумага типографская № 1.
Обыкновенная гарнитура. Высокая печать.
13,5 бум. л. Усл. печ. л. 27. Уч.-изд. л. 24,40
Изд. № 1/9745. Тираж 14.000 экз. Зак. 0774. Цена 2 р. 10 к.

Издательство «Мир»
Москва, 1-й Рижский пер., 2

Ордена Трудового Красного Знамени Московская типография № 7
Искра революции» Союзполиграфпрома при Государственном комитете СССР по делам издательств, полиграфии и книжной торговли. Москва, К-1, Трехпрудный пер., 9